DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENC/99-02

Multiple Comparison Pruning of Neural Networks

DISSERTATION

Donald Edward Duckro
Major, USAF

AFIT/DS/ENC/99-02

AFIT/DS/ENC/99-02

Multiple Comparison Pruning of Neural Networks

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Donald Edward Duckro, B.Ch.E., B.S.E.E., M.S.A.M.

Major, USAF

September, 1999

AFIT/DS/ENC/99-02

Multiple Comparison Pruning of Neural Networks

Donald Edward Duckro, B.Ch.E., B.S.E.E., M.S.A.M.

Major, USAF

Approved:

[illegible]	3 SEP 99
Dr. Dennis W. Quinn (Chairman)	Date
[illegible]	8/26/99
Dr. Thaddeus Tarpey	Date
Edward D White	8/26/99
Capt. Edward D. White	Date
Samuel J. Gardner III	8/24/99
Capt. Samuel J. Gardner III	Date
William E. Wiesel	8/26/99
Dr. William E. Wiesel (Dean's Representative)	Date

[illegible]

Robert A. Calico, Jr.
Dean

Acknowledgements

I thank my wife for her patience, understanding, and care. I thank my children for the joy they bring to me. Of course, my advisors: Dr. Quinn, Dr. Tarpey, Captain Gardner and Captain White have provided me enormous guidance. Additionally, my appreciation to Major John Crown for his expertise in goodness-in-fit and Weibull distributions.

Donald Edward Duckro

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol	Page
ψ growth function	11
d Vapnik and Chervonenkis dimension	11
α_i attentional strength	12
ρ relevance	12
H Hessian matrix	13
S selection matrix	14
δ mean delta of the output response	16
ε generalization error rate	17
t epoch or iteration of training	18
Λ_i, eigenvalue of the Hessian matrix	18
s_p Optimal Brain Damage salience measure	19
γ Lagrange multiplier	21
L_p Optimal Brain Surgery salience measure	22
α significance level	25
v degrees of freedom	25
q studentized range	25
$c_{ii}s^2$ estimated variance term	34
$c_{ij}s^2$ estimated covariance term	34
a Pareto shape parameter	37
k Pareto location parameter	38
g number of Bonferroni statements	41
β Weibull slope or shape parameter	46
ζ Weibull scale or characteristic life parameter	46
p-value probability value	48

AFIT/DS/ENC/99-02

Abstract

Reducing a neural network's complexity improves the ability of the network to be applied to future examples. Like an overfitted regression function, neural networks may miss their target because of the excessive degrees of freedom stored up in unnecessary parameters. Over the past decade, the subject of pruning networks has produced non-statistical algorithms like *Skeletonization*, *Optimal Brain Damage*, and *Optimal Brain Surgery* as methods to remove connections with the least salience. There are conflicting views as to whether more than one parameter can be removed at a time. The methods proposed in this research use statistical multiple comparison procedures to remove multiple parameters in the model when no significant difference exists. While computationally intensive, the Tukey-Kramer method compares well with *Optimal Brain Surgery* in pruning and network performance. When the Tukey-Kramer method has inefficient sampling requirements, Weibull distribution theory alleviates the computational burden of bootstrap resampling with single sample analysis, while maintaning comparable network performance.

Multiple Comparison Pruning of Neural Networks

I. Introduction

When presented with a data set for analysis, the engineer or statistician are both in the position similar to an artist presented with a canvas and palette of oils: the final representation is dependent upon the experience of the craftsman. Presuming that the data set contains the necessary information, either analyst would employ their appropriate algorithms "to extract and organize that information to obtain an accurate prediction rule" (12). It is the purpose of this research to observe the differing approaches of the engineer and statistician, and incorporate appropriate statistical discipline to the modification of neural networks to improve their accuracy.

Mathematics and statistics have supplied traditional data analysis techniques predating the computer, but the computer is not to be ignored. As the engineer has found the tool indispensable for machine learning and pattern recognition, so has the statistician refined the tools for classification and regression. Just as newer statistical techniques arise in principal component analysis, the engineer (connectionist) generates a stir in new developments of artificial intelligence and neural networks.

Unfortunately there is a lot of second-guessing between engineers and statisticians as to the effectiveness of the other's approach. Though many authors bridge the divide (10, 22, 45, 50, 62), Cherkassky (12) suggests that the differing fields achieve progress independently of one another. Such a view would imply that researchers of the various disciplines are entrenched or ill read. However, when the methodological developments are sufficiently diverse, the independent approaches that produce comparative performance in predictive capabilities only further reinforce the extractability of the underlying information within a data set. Both disciplines can value this fortune of order.

Cherkassky (12) recalls two great quotes from prominent statisticians to put perspective on the grandeur and limitation of statistics. The neural network community must consider B. Efron's quote, "Statistics has been the most successful information science, and those who ignore it are condemned to reinvent it." The statistical community must consider R.A. Fisher's quote, "It is the scientist, not the statistician, who constructs the structural model. It is the role of the statistician to study the inferential limitations of that model under various uncertainty mechanisms." Neither community can claim superiority in methodology, nor neither can afford to overestimate their role in information processing. As for the usefulness of artificial neural networks for statistical inference, the connectionist develops the network to address a specific problem with perhaps a large data set, a complex architecture, and a goal to produce adequate *generalization*—a measure of the network's ability to perform well on future examples (45). In contrast, the statistical theory would require less data within a given model to produce *interpretability.* In either case, successful inference is achieved as the data set grows, so the real criteria of appropriateness lies in performance on real, perhaps ill-posed problems with finite, maybe sparse data sets. "The best method should conform to the properties of the data at hand (12)."

Perhaps the greatest melding of the disciplines is occurring in network *pruning*, because the lessons of overfitted regressions apply to the neural networks as well. Overfitted networks pick-up the idiosyncrasies of the training data set, adversely affecting the capability to generalize (44). So, even though neural networks' overt size can be the key to quick learning, the truth of the Principle of Parsimony

> It may pay not to try to describe in the analysis the complexities that are really present in the situation (56)

has led the neural network community to search for simpler models. Employing the smallest possible number of parameters for adequate representation benefits the generalization goal of the network by removing the superfluous parameters that permit too much memorization of the training set (23). Thus, pruning algorithms have been developed to trim large, quick learning networks into sufficient, minimal networks. Typical algorithms reduce an oversized network in a step-by-step fashion similar to a statistical backward elimination procedure. Conversely, networks can also be built to an appropriate complexity using forward selection techniques analogous to those in statistics.

1.1 Scope

This research seeks to review the established links between statistical hypothesis testing and neural network pruning, and develop simpler hypothesis testing using the range, studentized range (24), Pareto (8, 37, 48), and Weibull (1) distributions for the pruning of neural network's features and architecture.

1.2 Dissertation Organization

Given the range of contributing resources, the table of symbols attempts to bridge the various symbol assignments made by different disciplines. In Chapter II, this dissertation presents background information on topics relevant to the research problem and related work that has already been accomplished in these areas. Chapter III describes the development of multiple comparison approach to neural network pruning and an analysis of the appropriateness of this new approach. Finally, Chapter IV summarizes the experimental results of multiple comparison pruning that lead to the conclusions of Chapter V.

II. Background

Neural networks have existed in nature for an exceptionally long time—since early animal life. However, the history of artificial neural networks is limited to the twentieth century. With the interchange of ideas between life scientists and engineers, the development of machines influenced by the theories of psychology and biology led way to Rosenblatt's *perceptron* (4, 46) in the late 1950s and the field of *artificial neural networks* in the mid-1980s (45). With these mathematical techniques established, artificial neural networks no longer required the original biological motivation. Although many authors will relate the machine to a biological counterpart, the machine is nothing more than a mathematical algorithm with computations activated by logic rules. The dynamics within the system appear to reflect intelligence, but only if the algorithm is prepared intelligently.

Beyond the inception of the perceptron, neural network methods continue to be developed, compared, and contrasted. Inasmuch as network algorithm preparation affects capability, the method of network construction is also under scrutiny. As mentioned in Chapter I, the scope of this effort is the application of untapped statistical tools for network size reduction. Rather than survey the wide field of neural network methods, a review of the feed-forward neural network will define some of the general terms that all neural networks share.

Neural networks are a wide class of flexible nonlinear models that are used for various purposes (regression, discrimination, data reduction, and nonlinear dynamical systems) (50). This dissertation is focused primarily with neural networks as predictors (regression, discrimination).

Figure 1 depicts the construction of the feed-forward neural network (9, 45) which is scaleable in complexity. The feed-forward network consists of hidden and output layers of units (also referred to as nodes, neurons, or perceptrons) and a layer of inputs. A bias is connected to each unit to provide an activation constant. One-way connections have associated weights that multiply the signal from the bias, inputs, and hidden units to their forward units. *Connectivity* is the description of how the lower layers and bias are connected to the upper layers. Full connectivity describes a network where each lower layer

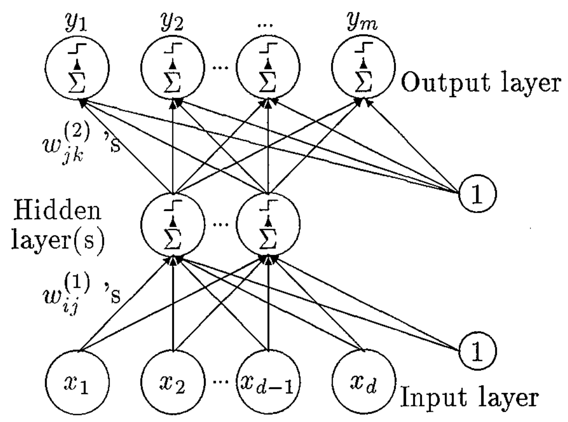


Figure 1. Feed-Forward Neural Network

unit is connected to each unit in the next layer. Reduced connectivity implies less than complete connections between subsequent layers. Network *architecture* is the description of existing layers, associated units and connectivity. The *complexity* of the network refers to the number of connections (free parameters) as a result of the network architecture.

Equation 1 is the general form of the network output

$$y_k = f_o \left(w_{0k}^{(2)} + \sum_{j=1}^{l} w_{jk}^{(2)} a_j \right) \tag{1}$$

where

$$a_j = f_h \left(w_{0j}^{(1)} + \sum_{i=1}^{d} w_{ij}^{(1)} x_i \right),$$

f_o is the activation function for the output layer, f_h is the activation function of the hidden layer, and a_j is the output of the jth hidden node which in turn becomes one of the inputs to the output layer. The network operates like a discrete thresholding device with each forward unit summing its weighted inputs and bias as an input to the unit's activation function, f_h or f_o. The activation function can be logistic sigmoid, hyperbolic tangent, or step. Otherwise, a linear activation function can be used to merely forward the sum of the weighted inputs.

The network weights w are randomly initialized with care so as to avoid saturating the hidden layer activation function. From a training data set $\mathbf{X}$, the weights are determined (*learned*) using an iterative algorithm designed to minimize an error function, $\mathcal{E}$. A commonly used training error function involves the sum of square error of the predicted versus target value. To facilitate computational analysis, the network weights are reorganized into a vector data structure as P single scripted elements of weight vector $\mathbf{w} = [w_1 \; w_2 \cdots w_P]$. The gradient of $\mathcal{E}(\mathbf{w})$ with respect to w describes the change in $\mathcal{E}(\mathbf{w})$ as w changes at a point $\mathbf{w}$. Finding $\mathbf{w}$ so that $\nabla\mathcal{E}(\mathbf{w}) = 0$ reveals a stationary point of $\mathcal{E}(\mathbf{w})$. Since the step function is not continuously differentiable, it is not acceptable as an activation function during learning. Widrow-Hoff's (4, 63) gradient descent is a classical weight update rule for a given learning rate η where for the *pth* weight

$$w_p \leftarrow w_p - \eta \frac{\partial \mathcal{E}}{\partial w_p}.$$

The weight updates represent the *back-propagation* of information into the network. The algorithm can either update the weights *on-line* after each training data vector $\mathbf{x} \in \mathbf{X}$ or after the *batch* of training vectors in $\mathbf{X}$ passes through the network. On-line training has the advantage of learning from every experience, faster convergence per data passed, and more likely avoiding local minima—but at a computational price. The iterations (*epochs*) continue until the stopping rule is achieved. The stopping rule may be tied to the training error goal, generalization on a validation set, or an order of epochs considered optimal for generalization (61).

Complexity, learning rates, training error and generalization have become comparative measures for neural networks in response to complex real-world problems that force network sizing growth to achieve successful neural network learning (33). Large neural networks are quick learners but may be less than adequate generalizers (performing poorly on future examples). Overfitted networks acquire the temperament of the training data set, negatively affecting the network's ability to generalize. This coincides with how statistical models are capable of success and failure in predicting future responses from empirical data by fitting or overfitting the data. Statistical relations undergo model reduction to avoid overfitting. Similarly, neural network complexity reduction prevents memorization.

So, even though neural network size can be the key to quick learning, a simpler model that fits the data may perform better as a predictor of outcomes for future examples. Model simplification involves removing nonessential weights, nodes, or features that may allow for better generalization. Thus, pruning algorithms have been developed to trim large, quick learning networks into sufficient, minimal networks. Network pruners like *Optimal Brain Surgery* (OBS) (25) improve generalization by trimming the excess weights from the network. To speed the pruning process, the results of Tukey (28, 40, 56, 57), Kramer (31) and Pareto (37, 48, 8) provide a statistical bridge between inference and decision theory for multiple comparison pruning. This proactive approach contrasts the more passive approach of model comparisons by Golden (22), Voung (59), and White (62) applying Wald and Wilks' mathematical methods for statistical inference on the suitability of an artificial neural network. The following sections discuss the theory used by the various pruning methods.

2.1 Statistical Inference

A review of the mathematical methods for statistical inference on the suitability of an artificial neural network includes the works of Efron (18), Wald (60), and Wilks (64) as applied by Golden (22), Paass (41), Voung (59), White (62). The theory is presented in this section and applications will be discussed in Section 2.3. Multiple comparison pruning will involve the works of Tukey (28, 40, 56, 57), Kramer (31) and Pareto (8, 37, 48) and the theory and application will be discussed in Chapter III.

2.1.1 Wilks Generalized Likelihood Ratio Test. The Wilks Generalized Likelihood Ratio Test (GLRT) procedure involves fitting the response function with both a full model of P parameters and a reduced model of M parameters, where $M < P$. Both models possess a model deviance that compares the log-likelihood of each fitted model to the log-likelihood of a perfectly fitted 'N' model with the number of parameters equal to N, the number of observations (40). The test investigates whether the full and reduced model deviances are significantly different. Since both model deviances have the 'N' model in common, only the likelihood values of the full and reduced model are necessary to complete

Neyman and Pearson's likelihood ratio statistic λ from a random sample $\mathbf{X}$

$$\lambda = \frac{L_{\Omega_o}(\mathbf{W})}{L_{\Omega}(\mathbf{W})} = \frac{\operatorname*{lub}_{\mathbf{W}\in\Omega_o} \prod_{i=1}^{N} \mathrm{f}(\mathbf{x}_i, \mathbf{W})}{\operatorname*{lub}_{\mathbf{W}\in\Omega} \prod_{i=1}^{N} \mathrm{f}(\mathbf{x}_i, \mathbf{W})},$$

where

$$\begin{aligned} \Omega &= \Re^P \\ \Omega_o &= \{\mathbf{W} \in \Re^P : W_{k_i} = 0, \quad i = 1, \cdots P - M, \quad i > j \Rightarrow k_i > k_j\}. \end{aligned}$$

Thus, the likelihood values are found by obtaining the maximum likelihood estimate (MLE) of the parameters for each model, and evaluating the respective likelihood functions. Wilks developed the GLRT based upon the χ^2 distribution's relevance as put forth in *Wilks Theorem* (64):

> If a population with a variate $\mathbf{x}$ is distributed according to the probability function $\mathrm{f}(\mathbf{x}, W_1, \cdots, W_P)$, such that optimum estimates w_i of W_i exist which are distributed in large samples according to the probability density function
>
> $$(2\pi)^{-\frac{P}{2}} \left| -E \frac{\partial^2 \log \mathrm{f}}{\partial W_i \partial W_j} \right|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \sum_{i,j=1}^{P} - E \frac{\partial^2 \log \mathrm{f}}{\partial W_i \partial W_j} (w_i - W_i)(w_j - W_j) N \right) (1 + \phi),$$
>
> where ϕ is of order $N^{-\frac{1}{2}}$; then when the null hypothesis $H : W_i = W_{oi}$, for $i = M+1, \cdots, P$, is true, the distribution of $X^2 = -2 \log \lambda$ is, except for terms of order $N^{-\frac{1}{2}}$, distributed like χ^2 with $P - M$ degrees of freedom.
> Note: The norm $|\bullet|$ results in a scalar independent of i and j.

In testing a full and reduced model, all $W_{oi} = 0$. So the X^2 test statistic is $-2 \log$ of the ratio of likelihood values, or simply the difference in the two model deviances. If the test statistic is less than the critical value χ^2_{P-M}, that is the chi square distribution with $r = P - M$ degrees of freedom at a specified significance level, then fail to reject that the null hypothesis H is true.

The application of the GLRT on neural networks includes testing the null hypothesis that a particular input has no affect on the network performance, that is all of the connection weights of an input converge to zero as the sample size (training data) increases (22). In other words, given a full model, the reduced model eliminates the questionable input and trains consistent with the full model. A GLRT between the full and reduced models provides a decision as to the significance of the input in question. GLRT applications are further discussed in Section 2.3.

2.1.2 Wald Test. Wilks found the limit distribution of the test statistic, λ, to be χ_r^2 if the hypothesis to be tested was true. As a follow-up to Wilks GLRT's limiting central χ^2 distribution under the null hypothesis, Wald proposed a limiting noncentral χ^2 distribution under sequences of local alternatives (59). Wald (60) concluded the following:

> If the true parameter point $\mathbf{W}$ is not an element of Ω_o, the distribution of the statistic $-2\log\lambda$ approaches the distribution of a sum of noncentral squares
>
> $$U^2 = u_1^2 + \cdots + u_r^2$$
>
> where the variates $u_1, \cdots, u_r$ are independently and normally distributed with unit variances and
>
> $$\lambda^2(\mathbf{W}) = \sum_{i=1}^{r} (Eu_i)^2.$$

The Wald test allows for the significance testing of a subset of parameters within a given model. The application of the Wald Test on neural networks includes testing a collection or linear combination of connection weights within a particular neural network to determine if they have no affect on the network performance. Setting the connection weights to zero, the reduced model can be compared with the full model using Wilks GLRT. Wald Test applications are further discussed in Section 2.3.

2.1.3 Efron Bootstrap Resampling Plan. The Bootstrap Resampling Plan was first proposed by Efron in 1979 as an approach to describe the sampling distribution for the estimator using the true cumulative distribution function (cdf), $\mathcal{F}$. If $\mathcal{F}$ relates to the empirical cdf $\hat{F}_N$ as $\hat{F}_N$ relates to a secondary sample drawn from itself, then $\mathcal{F}_N$, the sampling (bootstrap) distribution of the estimate under $\hat{F}_N$, can be used as a good approximation of the sampling distribution for the estimate under $\mathcal{F}$ (41). Efron (18) provides a three step Monte Carlo approximation:

Let

$$R(\mathbf{X}, \mathcal{F})$$

be a random variable of interest, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ indicates the entire independent, identically distributed (iid) sample $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ from $\mathcal{F}$. On the basis of having observed $\mathbf{X}$ we wish to estimate some aspect of R's distribution, $\mathtt{STAT}_{\mathcal{F}}R$.

1. Fit the nonparametric MLE of $\mathcal{F}$,

$$\hat{F} : \text{mass } \frac{1}{N} \text{ at } \mathbf{x}_i, \quad i = 1, 2, \cdots, N.$$

2. Draw a 'bootstrap sample' from $\hat{F}$,

$$\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_N^* \overset{iid}{\sim} \hat{F},$$

   and calculate $R^* = R(\mathbf{X}^*, \hat{F})$.

3. Independently repeat step 2 a large number of times (B) obtaining 'bootstrap replications' $R^{*1}, R^{*2}, \cdots, R^{*B}$, and calculate $\mathtt{STAT}_* R^*$, the bootstrap estimate of $\mathtt{STAT}_{\mathcal{F}}R$.

The applications of the Bootstrap Resampling Plan on neural networks include approximating the sampling distribution of the prediction, estimating parameters of that distribution, and any possible bias. Bootstrap applications are further discussed in Section 2.3.

2.2 Salience Measures

The definition of salience is "a pronounced trait". For neural networks a salience measure determines the effect a weight has on the training error (33). A review of the salience measures for the determination of redundancy or relevancy of weights and nodes of an artificial neural network includes the theory of Akaike (3) or Vapnik and Chervonenkis (58) as applied by Baum and Haussler (6), Hassibi (25), Le Cun (33), Levin (35), Mozer (38) et al. The tools are presented in this section and applications will be discussed in Sections 2.4 and 2.5.

2.2.1 Complexity Measures. Network complexity (or size) affects both the speed of learning and the number possible algorithms (the set of $\mathcal{Y}$ hypotheses). Given a network with free weights to learn a concept or true input-output relationship $\mathbf{z}$, the learning algorithm ends at hypothesis $\mathbf{y} \in \mathcal{Y}$. Reduced training error depends on a final $\mathbf{y}$ that has a large rate of success $\nu_{\mathrm{y}} = \frac{\mathrm{n}}{N}$, where n is the number of successful predictions out of N observations. However, generalization may be poor without uniform convergence of ν_{y} to the population probability of success Π_{y} of $\{\mathbf{x} \in \mathcal{X} : \mathbf{y}(\mathbf{x}) = \mathbf{z}(\mathbf{x})\}$. The condition of uniform convergence (2, 58) and hence good generalization follows:

For any $\epsilon > 0$,

$$Pr\left[\sup_{\mathrm{y}\in\mathcal{Y}} |\nu_{\mathrm{y}} - \Pi_{\mathrm{y}}| > \epsilon\right] \leq 4\psi(2N)e^{-\epsilon^2 N/8}. \tag{2}$$

Inequality 2 will have a small right hand side for large N since $e^{-\epsilon^2 N/8}$ is exponentially decaying in N—provided the growth function $\psi(2N)$ grows slowly in comparison. The growth function ψ measures the maximum number of different binary functions covered by $\mathcal{Y}$; thus, $\psi(N) \leq 2^N$. Inasmuch as $\mathcal{Y}$ encompasses the characteristics of the training data, Vapnik and Chervonenkis developed the complexity measure known as the VC dimension, d, as the smallest N at which $\mathcal{Y}$ starts failing to induce all possible 2^N binary functions of any N samples. In such a case (2, 58), $\psi(N) \neq 2^N$, but is bound by a polynomial $N^{\mathrm{d}}+1 \geq \psi(N)$. The VC dimension d guarantees that the growth function will be dominated by the negative exponential; thus, producing a small right hand side to Inequality 2.

The VC dimension, as a measure of the network complexity, permits inference on the generalization capability of the network. The VC dimension is closely related to the number of weights (6) with generalization expected for $\mathrm{d} \ll N$. The VC dimension as used for sizing and complexity measures is discussed in Section 2.4.

Another complexity measure is An Information Criterion (AIC):

$$\mathrm{AIC}(\mathrm{w}) = 2P - 2\log(L(\mathrm{w})),$$

where P is the number of independently adjustable parameters and $L(\mathrm{w})$ is the maximum likelihood. Akaike (3) suggests a minimum AIC estimate for statistical model selection is equivalent to a hypothesis test without the significance level decision. An elementary look by Svarer, et al. (54) of Akaike's estimate of the generalization error or Final Prediction Error (FPE):

$$\widehat{\mathrm{FPE}}(P) = \left(\frac{N+P}{N-P}\right)\mathrm{MSE}_{\mathrm{train}}$$

highlights the expected rise in generalization error as the number of parameters P approaches the number of training samples N. Svarer, et al. simplify the right-hand side with the training set mean square error, $\mathrm{MSE}_{\mathrm{train}}$, in lieu of Akaike's cost function. Wang, et al. (61) incorporate FPE and $\mathrm{MSE}_{\mathrm{train}}$ into a criterion for optimal stopping time and network size as discussed in Section 2.4

2.2.2 Differential Measures. In order to perform salience investigations of the weights, the affect on training set error $\mathcal{E}$ must be measurable. Several approaches are used to measure the differential affect of $\mathcal{E}$. These measures become the basis of decision for salience strategies.

Each network node has an associated attentional strength, α_i, denoting the level of activity from the node. The attentional strength is equal to zero if the node has no influence and unity if the node is fully active. The relevance, ρ, of the network node as defined by Mozer, et al. (38):

$$\rho_i = \mathcal{E}_{without\ unit\ i} - \mathcal{E}_{with\ unit\ i},$$

is approximated by

$$\hat{\rho}_i = -\frac{\partial \mathcal{E}}{\partial \alpha_i}.$$

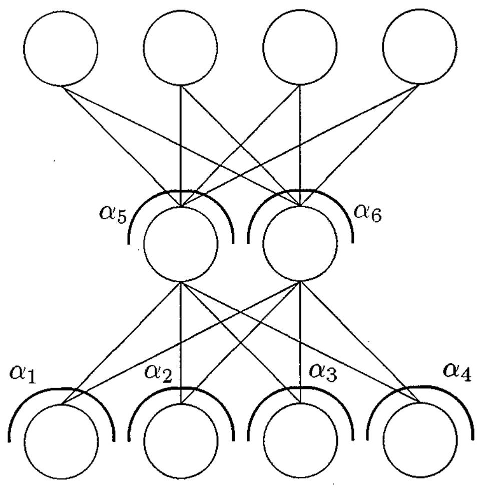


Figure 2. Attentional Strength Coefficients of Network Units

Although α_i is not a parameter of the system, but merely notational convenience to estimate relevance, Figure 2 pictorially illustrates the attentional strength as a coefficient controlling the emphasis of each unit. The application of relevance is further discussed in Section 2.5.

Another approach considers a local model of $\mathcal{E}$ as a function of a parameter vector $\mathbf{W}$. The Hessian matrix $\mathbf{H}$ consists of elements h_{ij} defined by

$$h_{ij} = \frac{\partial^2 \mathcal{E}}{\partial w_i \partial w_j},$$

and the gradient of $\mathcal{E}$ with respect to $\mathbf{W}$ produces elements

$$g_i = \frac{\partial \mathcal{E}}{\partial w_i}.$$

If training has converged, then $\mathcal{E}$ is at a local minimum and the gradient is negligible in a Taylor series expansion of the perturbation of $\mathcal{E}$, shifting focus to the second partial derivative to analytically predict the effect of perturbing the parameter vector (33). The methods for using differential measures are discussed in Section 2.5 and used in Chapters III and IV.

2.2.3 Principal Component Analysis. Given the large dimensionality of neural network's weight-space, a technique for reducing dimensionality and the variance of prediction can be as effective as pruning nodes and connections. Principal component analysis can identify the least salient eigen-nodes and their effect on the output error. The removal of those eigen-nodes reduces dimensionality. Although there is no guarantee that low variance variables have little effect on the error, principal components with low variances may only be reflecting the noise of the process and are candidates for elimination. By alleviating the model of these components, the response variance could decrease. The cost of dimensionality reduction for the benefit of generalization is an increase in model bias affecting the training set error.

If principal components are used to preprocess the data, the singular value decomposition of the training set data matrix will produce the orthogonal matrix necessary to calculate the linear combinations of the data that describe the most variability. However, since system inputs involve various units of measure, the principal component will depend heavily on the units used. For a natural problem, the first principal component may only be a reflection of the overall size of the system (45). Such a dimension may not be a discriminating factor. In order to transcend units, *Principal Component Pruning* (35) uses the eigendecomposition of the correlation matrix, akin to rescaling the data to unit variance before calculating the covariance matrix. Section 2.5 further discusses this approach.

2.3 Statistical Network Evaluation

From Section 2.1, the work of Wilks, Wald, and Efron provided statistical tools necessary to infer the appropriateness of model selection. The following sections review published applications of these tools.

2.3.1 Analysis of Hidden Unit Representation. White (62) reviews statistical inference on network architecture with the classic irrelevant hidden unit hypothesis

$$H_o : \mathbf{SW} = 0,$$

where **S** is the selection matrix for weights associated with the units under consideration for relevancy. For large N and a normal limiting distribution of weights, the χ^2 distributed test statistic in comparison to a critical value provides the decision rule for the hypothesis.

However, when H_o is true, the weights into the irrelevant unit(s) are not locally unique (i.e. have no effect on the network output), and have a limiting mixed Gaussian distribution. Even though the resulting test statistic is no longer χ^2 distributed, Wald provides statistical tools discussed in Section 2.1 to obtain a test statistic asymptotically distributed χ^2_r (21).

The Wald Test can be further expanded to examine hypotheses that a subset of weights of a neural network are equal to zero. Rather than test all weights to or from a unit, a general subset of weights amongst various units can be tested for their proximity to zero. Additionally, linear combinations of weights can be tested for a summary net effect. However, testing for zero may be misleading. Connection relevance is more a product of the weight and the signal rather than the weight itself. Nonetheless, the Wald Test provides analysis on a particular network without the need for retraining.

2.3.2 Bootstrap Estimates of Predictive Distribution. Paas (41) demonstrates the bootstrap algorithm on a small neural network to obtain confidence intervals for the weight parameters. By resampling with replacement, numerous bootstrap data sets are used to train the small network and obtain a histogram of the weight estimates. Confidence intervals can be drawn from the empirical cdf of the bootstrap estimates which approximates the sampling distribution of the underlying cdf. Additionally, for each bootstrap weight vector estimate, the complete set of input vectors can be applied to the network to obtain a predictive distribution of the output.

2.3.3 Bootstrap Approach for Relative Effects of Inputs. In contrast to Paas' work in sampling the input-output pairs of the original sample in an *unconditional* bootstrap to observe the underlying distribution, Baxt and White (7) use *conditional* bootstrap to evaluate the inputs of a neural network predicting myocardial infarction. Conditional bootstrap avoids making inferences on the general population by using residual sampling to take into account the input patterns actually observed. The input patterns are held fixed, while the output is perturbed in such a way that the probabilistic relation between input and output is upheld. For their study, they perturb the output toggle for infarction/no infarction by comparing the conditional probability of the target given the input vector with a uniform $[0, 1]$ random variable independent of the input. The output is assigned positive when the conditional probability exceeds the uniform random variable. Each input-perturbated output set represents a bootstrap sample. Resampling provides a means of making inferences on the sample population.

Training the neural network with the original sample establishes weights for the network. The original sample mean deltas δ_i^S are computed for each input parameter by varying each input vector component individually and measuring the output response. Using a bootstrap sample and retraining, similar network *pseudosample* mean deltas are calculated. After B resamplings, a histogram and average of the pseudosample mean deltas for the bootstrap mean deltas δ_i^B of each input are obtained. If there exists a bias between the bootstrap and original sample mean deltas, then the same bias exists between the original sample mean delta and true delta δ_i^T. The following equality expresses whether the original sample over/underestimates the true impact and can be used to solve for the expected true delta in the output function for each input parameter:

$$\delta_i^B - \delta_i^S = \delta_i^S - \delta_i^T.$$

The bias adjusted confidence histograms can be used to determine whether the input parameters impact on the output is significantly different than zero and thus consequential.

2.3.4 T-test for Feature and Model Selection. Steppe and Bauer (53) combined much of the aforementioned statistical methods for neural network evaluation. The resulting comprehensive selection method comprises three stages for network determination. The stages include an initial architecture selection algorithm, a salience screening procedure, and a feature selection algorithm. Within the feature selection algorithm, addition architecture refinements are performed.

- **Initial Architecture Selection.** The GLRT for full and reduced models determines the appropriate number of hidden nodes.
- **Saliency Screening.** In a departure from all previous observed salience evaluations, Steppe uses a gradient method for saliences and the Bonferroni t-test to identify *noise-like* features as irrelevant to the network performance.

- **Feature Selection.** The GLRT for full and reduced models determines the appropriateness of hidden and input nodes serially.

This process integrates several statistical tools into a complete neural network evaluation procedure. Because of the serial implementation, the required training is intensive. Further research is being done to incorporate parallel processing.

2.4 Optimal Network Sizing

From Section 2.2, the work of Akaike, Vapnik and Chervonenkis provided complexity criteria necessary to infer the appropriate network size. The following sections review published positions on suitable network sizing.

2.4.1 Sample Versus Network Size for Generalization. Baum and Haussler (6) developed bounds for sample size as an order of network complexity and generalization goal. With l linear threshold nodes in the hidden layer, a generalization error rate $\varepsilon \leq \frac{1}{8}$ is obtainable for a sample size greater than an order function O

$$N \geq O(\frac{P}{\varepsilon} \log \frac{l}{\varepsilon}),$$

dependent on a training success rate

$$\nu_{\mathbf{y}} = (1 - \frac{\varepsilon}{2}).$$

Conversely, generalization will fail for sample size less than an order function O

$$N < O(\frac{P}{\varepsilon}).$$

Finally, they propose that for any number of hidden layers the VC dimension is bounded

$$\mathrm{d} \leq 2P \log_2(el)$$

where e is the base of the natural logarithm. Baum and Haussler are joined by others in the continued pursuit of bounds on N and d based upon different network activation functions and architecture. Bounding conclusions become stopping points for pruning and construction algorithms.

2.4.2 Effective Machine Complexity. Wang et al. (61) considers the generalization performance as a function of the learning process for a given network complexity. Within the learning process there are phases of generalization performance with the optimum occurring before the training data is learned to a global minimum error. During the learning process, the network performs with an effective size smaller than the VC dimension d. Combining the two progressions, the goal is to uncover the smaller effective size at an iteration of training (epoch), t, that produces better generalization performance.

Given $\mathbf{W}$ as the parameter vector globally minimizing the error function $\mathcal{E}(\mathbf{w})$, then $\mathbf{H}$ is nonsingular with eigenvalues $\Lambda_1, \cdots, \Lambda_i, \cdots, \Lambda_P$. The effective size of the network at epoch t is

$$\mathrm{d}(t) = \sum_{i=1}^{P} \left[1 - (1 - \eta\Lambda_i)^t\right]^2 . \tag{3}$$

Equation 3 reflects the network capacity used at t. As t grows large, Equation 3 increases to d monotonically.

The asymptotically unbiased estimate of generalization error as a function of epochs and complexity is

$$\widehat{\mathrm{FPE}}(\mathbf{w}_t) = \mathrm{MSE}_{\mathrm{train}} + \frac{2\sigma^2}{N} \sum_{i=1}^{P} \left[1 - (1 - \eta\Lambda_i)^t\right] ,$$

when σ^2 is known. The criterion for finding the optimal stopping time and network size follows

$$\min\{\widehat{\mathrm{FPE}}(\mathbf{w}_t) : P, t = 1, 2, \cdots\}.$$

Wang's approach requires the global minimization of the network and the sort of all interim parameter vectors. However, it does extend the AIC by incorporating a stopping time to the network sizing criterion.

2.5 Network Pruning

From Section 2.2, the work of Akaike, Vapnik and Chervonenkis provided complexity criteria necessary to support network reduction by salience measures. The following sections review popular network reducing pruning algorithms.

2.5.1 Skeletonization.

Mozer et al. (38) consider the *relevance* of individual units in a network as a salience measure. The most relevant units are critical to performance and form the *skeleton* version of the network. Relevance must not be confused with magnitude of weights associated with a unit; but rather, relevance is the difference in training error with and without the unit. To determine relevance for l hidden and d input units, the computational cost is of order $O\left((l+d)\cdot P\right)$. So an approximation of relevance is derived as

$$\hat{\rho}_i = -\frac{\partial \mathcal{E}}{\partial \alpha_i},$$

where $0 \leq \alpha_i \leq 1$ is the attentional strength of the unit ranging from zero to full influence, as defined by Mozer et al. (38). The derivation is based on the definition of relevance as the difference in error with and without the node, and the derivative of the error with respect to the attentional strength at unity as it holds approximately at zero. When the output pattern is close to the target, a better estimate of relevance is calculated based on an error function, $\mathcal{E}_N = \sum |z_k - y_k|$, as opposed to the sum of square errors.

The procedure separates training from pruning. After training the network, the procedure computes the relevance $\hat{\rho}$ for each unit and removes the unit with the smallest $\hat{\rho}$. The procedure repeats the training, relevancy determination, and deletion until a specified complexity level (or VC dimension) is reached.

2.5.2 Optimal Brain Damage.

In contrast to *Skeletonization*, Le Cun et al. (33) prune the neural network by removing unimportant weights as opposed to irrelevant nodes. The strategy of *Optimal Brain Damage* (OBD) is to delete the weights with the smallest salience, in this case, the least effect on the training error. As discussed in Section 2.2, the salience measures s_p are computed from the second derivative of the error function with respect to the weights. Recall for the Taylor series of the error perturbation:

$$\delta\mathcal{E} = \sum_{p=1}^{P} \frac{\partial \mathcal{E}}{\partial w_p}\delta w_p + \frac{1}{2}\sum_{p=1}^{P} \frac{\partial^2 \mathcal{E}}{\partial w_p^2}\delta w_p^2 + \frac{1}{2}\sum_{q=1}^{P}\sum_{p=1,p\neq q}^{P} \frac{\partial^2 \mathcal{E}}{\partial w_p \partial w_q}\delta w_p \delta w_q + O(\| \delta \mathbf{w} \|^3),$$

the local minimum assumption deleted the first derivative terms and the quadratic assumption deleted higher order terms. A diagonal approximation without cross terms further simplifies the Taylor series with the assumption that the perturbation of the error from the deletion of several weights is the sum of the saliences caused by the deletion of each weight individually ($\delta w_q = 0$ for all q except $\delta w_p = -w_p$, see (23)). The diagonal terms of the Hessian matrix $\mathbf{H}$ of the objective error function $\mathcal{E}$ are

$$h_{pp} = \frac{\partial^2 \mathcal{E}}{\partial w_p^2} = \frac{\partial^2 \mathcal{E}}{\partial a_j^2} x_i^2,$$

where $a_j = \sum_{i \to j} w_{ij} x_i$ is the weighted sum input into a node activation function. Hence, the remaining terms within the Taylor series approximation are the individual saliences

$$\mathrm{s}_p = \frac{1}{2} h_{pp} w_p^2.$$

The second derivatives are backpropagated from layer to layer starting from the boundary condition of the output layer. An approximation corresponding to the Levenberg-Marquardt (33, 45) simplifies the second derivative with respect to the last layer of weighted sums

$$\frac{\partial^2 \mathcal{E}}{\partial a_k^2} = 2 f_o'(a_k)^2$$

and to the hidden layer(s)

$$\frac{\partial^2 \mathcal{E}}{\partial a_j^2} = f_h'(a_j)^2 \sum_{j \to k} w_{jk}^2 \frac{\partial^2 \mathcal{E}}{\partial a_k^2}.$$

Thus, the reduced Taylor series approximation of the set of saliences can be found by the following pair of computational equations:

Into the Output Layer

$$s_{jk} = f_o'(a_k)^2 x_j^2 w_{jk}^2,$$

Into the Hidden Layer(s)

$$s_{ij} = f_h'(a_j)^2 \left(\sum_{j \to k} w_{jk}^2 f_o'(a_k)^2 \right) x_i^2 w_{ij}^2.$$

The OBD procedure runs similar to *Skeletonization.* After training a reasonable network, compute the salience for each weight and zero the weights with the smallest saliences. Repeat the training, salience determination, and zeroing until a specified complexity level (or VC dimension) is reached.

2.5.3 Optimal Brain Surgery. In an analysis of OBD, Hassibi and Stork (25) find the Hessian diagonalization for OBD to be inappropriate for the strongly non-diagonal problems they encounter. The diagonalization assumption causes the elimination of the wrong weights, so Hassibi and Stork consider the inverse of the full Hessian $\mathbf{H}$ in determining the salience of the weights subject to pruning. If "Damage" is descriptive of the former approach, then *Optimal Brain Surgery* (OBS) suggests the corrective action necessary for the dominant off diagonal terms. OBS uses the same Taylor series approximation for the error function. If γ is the Lagrange multiplier of the Lagrangian of the second order term and zeroed weight constraint

$$\mathrm{L} = \frac{1}{2}\Delta\mathbf{w}^T \cdot \mathbf{H} \cdot \Delta\mathbf{w} + \gamma(\mathbf{e}_p^T \cdot \Delta\mathbf{w} + w_p),$$

where $\mathbf{e}_p$ is the unit vector in weight space corresponding to weight w_p, then the optimal weight change is

$$\Delta\mathbf{w} = -w_p \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}^{-1} \cdot \mathbf{e}_p. \tag{4}$$

The resulting OBS saliences for all the weights

$$\mathrm{L}_p = \frac{1}{2}\frac{1}{[\mathbf{H}^{-1}]_{pp}} w_p^2 \tag{5}$$

are consistent with OBD for true diagonal Hessians.

The OBS procedure runs similar to OBD. After training a reasonable network, compute $\mathbf{H}^{-1}$ and then the salience L_p for each weight. For the smallest salience of Equation 5, use the corresponding weight into Equation 4 to calculate the optimal weight change for all other weights in the network, precluding the need to retrain. Recompute $\mathbf{H}^{-1}$, saliences, and weight updates in an iterative fashion until a stopping criterion (i.e. $\max \Delta E, \min \mathrm{d}, \min r$) is reached.

In contrast to OBD, OBS considers the calculation of P^2 elements of the $\mathbf{H}$ and its inverse, $\mathbf{H}^{-1}$. Fortunately, the Hessian can be reduced to a covariance matrix form, simplifying the computational cost and allowing a recursive formula for computing the inverse. Thus, OBD's assumption of a diagonal Hessian can be avoided with OBS, without great expense, especially since network retraining is not necessary for OBS.

Hassibi et al. (26) attempt to solve the computational burden for larger networks with a dominant eigenspace decomposition of the inverse Hessian from the signal processing experience. As with any other approximation of the Hessian, the pruning technique will perform below the capability using the true Hessian. Unfortunately, their study reported excessively poor performance for eigenspace decomposition leaving the computational burden unresolved.

2.5.4 Pruning by Principal Component Analysis (PCA). In contrast to the previous pruning algorithms, *Principal Component Pruning* (PCP) by Levin et al. (35) does not require the calculation of the full Hessian matrix nor the retraining of the network. Instead, the weight and node activity correlation matrices of each layer are calculated. Through eigenspace decomposition, the principal components of the layer activity can be ranked. The deletion of eigennodes least affecting the validation error reduces the dimensionality of the network, or *effective* number of parameters. The weights of the layer are projected onto a subspace of fewer eigenvectors until a stopping criterion is reached.

PCP does not eliminate actual weights like OBD and OBS. The computational aspects of the network for generalization remain intact. However, PCP reduces the effective level of complexity to improve generalization and the technique does not require the computational burden of retraining and full Hessian analysis. Comparing the PCP approach to methods developed in this dissertation may be the basis of future work.

2.6 Chapter Summary

In review, the tools for statistical inference can be used to evaluate the neural networks in search of simpler models. Complexity measures help define the optimal sizing and training for networks. The complexity measures provide support for network reduction. Within network reduction approaches, salience measures simplify networks by eliminating weights or nodes that are inconsequential with respect to the error function. In Chapter III, the research joins the statistical rigor of multiple comparison procedures with the computational pruning algorithms in an attempt to intelligently remove inconsequential weights in a more efficient manner.

III. Multiple Comparison Pruning

This chapter develops a new method of implementing simultaneous inference procedures to accelerate the elimination of network connections (pruning) with statistical discipline.

3.1 Simultaneous Inference Procedures

A common theme amongst pruning algorithms is the role of salience measures in determining which weights are important. Weight salience is a measure of the effect each weight has on the training error. These saliences may be derived from a Taylor series approximation of the perturbation of the objective error function $\mathcal{E}(\mathrm{w})$. Weights with small saliences have little affect on error and are subject to pruning. However, obtaining saliences sometimes may be computationally burdensome; for instance, when inverting the Hessian matrix as seen in Equation 5. The efficiency of parameter elimination becomes an issue for large networks posing a formidable task.

Parameter elimination can be based upon minimal, significantly less, or neighboring salience. Shortsighted algorithms may consider prescribed decision rules that ignore the significance between saliences. Given multiple saliences for the parameters in question, multiple comparisons may improve the decision making steps within the pruning algorithm. Rather than eliminate parameters one at a time, comparatively small saliences can be grouped and eliminated in batch. Batch elimination can reduce the number of loops, retraining, or Hessian matrix inversions in a pruning algorithm.

If multiple comparisons are of interest, conservative approaches include methods by Bonferroni, Scheffé, Holm, and Tukey. The more prominent Bonferroni multiple comparison procedure is appropriate when a particular set of pairwise comparisons, contrasts, or linear combinations of saliences are specified in advance. However when increasing the number of comparisons of interest, the confidence intervals are narrowed with the Tukey procedure. The Tukey multiple comparison procedure is sometimes referred to as *honestly significant difference tests* (40). That bold assertion reflects the envelopment of all pairwise comparison tests and thus the ability to data snoop naturally without affecting the confidence coefficient or significance level. The Scheffé procedure also permits data snooping in that it accounts for all possible contrasts as opposed to pairwise comparisons. The Holm procedure is a computationally complex refinement of the Bonferroni procedure that updates the significance level as the number of comparisons are accepted as significantly different.

3.2 Tukey Multiple Comparison Procedure

The *Tukey Multiple Comparison Procedure* (MCP) considers the set of all pairwise comparisons:

$$\begin{aligned} H_0 &: \quad \mu_i = \mu_j \\ H_a &: \quad \mu_i \neq \mu_j \end{aligned} \tag{6}$$

where $i, j = 1, 2, \cdots, P$ with $i \neq j$. The family significance level is exactly α when all the sample sizes n_i are equal. When the sample sizes are not equal, the procedure is sometimes called the *Tukey-Kramer procedure* (40) and the family significance level is less than α. Thus, the procedure is conservative for nonequal sample sizes.

The Tukey procedure uses the studentized range distribution. For P observations $\mathrm{L}_1, \cdots, \mathrm{L}_P$ independently distributed $N(\mu, \sigma^2)$, suppose the variance estimate s^2 is based on v degrees of freedom independent of L_i, then the resulting studentized range, q, for salience is:

$$q = \frac{\max \mathrm{L}_i - \min \mathrm{L}_i}{s}.$$

For each pairwise comparison, the test statistic used for Hypothesis 6 is:

$$q^* = \frac{\sqrt{2}\,(\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j)}{s(\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j)} = \frac{\sqrt{2}\,(\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j)}{\sqrt{\mathrm{MSE} \cdot \left(\frac{1}{n_i} + \frac{1}{n_j}\right)}} \tag{7}$$

and H_0 is rejected when $|q^*| > q(1 - \alpha; P; v)$, thereby concluding a significant difference between the pair. With equal sample size n for all L_i, then $v = nP - P$. Otherwise, if L_i have nonequal sample sizes, then $v = \sum_{i=1}^{P} n_i \; - P$.

3.3 Bootstrap Training

As discussed in Chapter II, Efron (18) has shown that the Bootstrap Algorithm finds a distribution that can be used as a good approximation of the sampling distribution for the estimate under the true cdf. However, the notion of retraining with B samples is as tedious as the numerous cycles through a pruning algorithm. With the resampling incorporated into the network training algorithm, the weight estimates bootstrap about a minimum to approximate the sampling distribution. The weight estimates are obtained by retraining from a constant set of intermediate weights determined by an initial training session of the complete training data set. Only one cycle produces the multiple estimates needed to proceed with the simultaneous inference procedure.

With 128 bootstrap replications of training data sets, the histograms in Figures 3 and 4 illustrate a resulting weight and salience estimate distribution. The Anderson-Darling goodness-of-fit test fails to reject normality at level 0.1. Thus, resampling the available data set produces the necessary samples for inference of the means.

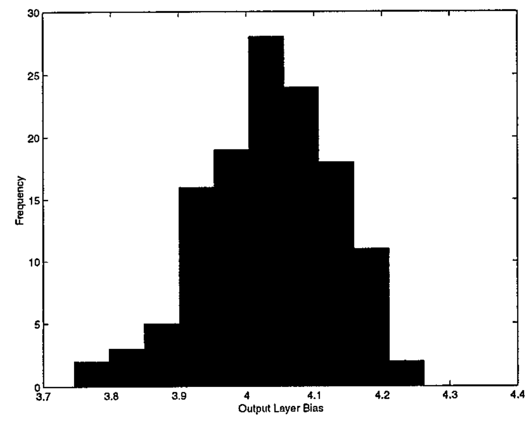


Figure 3. Bootstrap Distribution of Weight: Large Sampling, Weights of 1st Result Becomes Constant Initial Weights

With the Bootstrap Algorithm (Section 2.1.3) nested within the network training algorithm, an appropriate number of estimates can be collected to produce the mean weights

$$\bar{w}_p = \frac{\sum_{b=1}^{B} w_p^{*(b)}}{B}$$

and mean saliences

$$\bar{L}_p = \frac{\sum_{b=1}^{B} L_p^{*(b)}}{B},$$

where $w_p^{*(b)}$ and $L_p^{*(b)}$ are the p*th* weight and salience from the b*th* bootstrap.

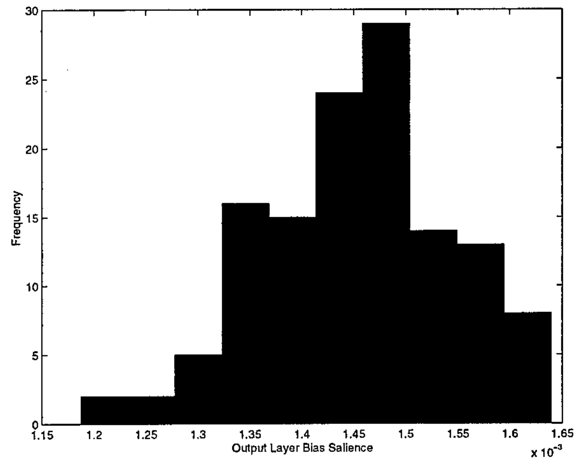


Figure 4. Bootstrap Distribution of Salience: Large Sampling, Weights of 1st Result Becomes Constant Initial Weights

For a pruning algorithm's parameter set under review, a comparative approach on the means of the parameter set can be performed using Tukey MCP. The ability to make simultaneous inferences is a worthy modification to acceptable pruning algorithms. The performance of the modified pruning algorithm can be compared with its root form for computational speed, training error, and generalization. Section 3.4 presents an application of multiple comparison pruning using the benchmark Monk's Problems.

3.4 Example: Monk's Problems

The Monk's Problems (55) are excellent benchmark Boolean tests for pruning algorithms. Monk's problems concern the classification of robots exhibiting six different features. Each feature has either two, three, or four possibilities as follows:

1. head shape $\in$ round, square, octagon
2. body shape $\in$ round, square, octagon

3. is smiling ∈ yes, no

4. is holding ∈ sword, balloon, flag

5. coat color ∈ red, yellow, green, blue

6. w/ necktie ∈ yes, no.

These features can describe 432 different robots. The three problems proposed by Thrun (55) involve the training of a neural network to discern certain distinctions of the robots to include:

1. head shape and body shape are the same, or coat color is red

2. two of the six features have the first value: round, yes, sword, red

3. holding a sword and coat color is red, or coat color isn't blue and body shape isn't octagon.

The Monk's Problems randomly draw from the 432 legal examples. The training set sizes of the three problems are set at 124, 169, and 122 respectively. The third problem is corrupted with a 5% error in the training set. For the purposes of illustration the first Monk's problem is used to demonstrate Tukey MCP.

For the first Monk's Problem, the neural network has 17 on/off inputs, one for each of the features' various possibilities as illustrated in Figure 5. The network begins with

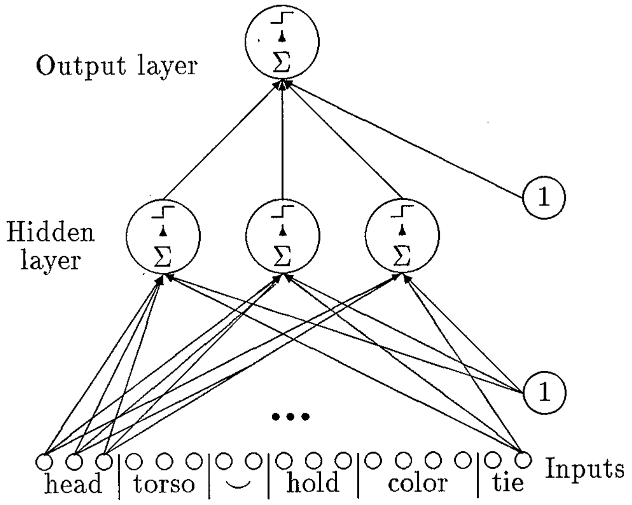


Figure 5. Neural Network for Monk's Problems

58 connections between the input layer, the three hidden nodes, and the one output node. Minimally, the pruning algorithm is expected to detect which inputs are important to discern the desirable characteristics. The pruning algorithm may test the reduced network with the entire 432 legal examples.

3.4.1 Multiple Weight Comparison Pruning. For multiple comparison pruning, a random sample of 124 robots is drawn from the 432 legal examples. Following Efron's approach (18), a bootstrap sample of 124 examples is drawn *with replacement* from the random sample of 124 robots. Certain elements of the random sample may appear more than once or not at all in the bootstrap sample. The network is trained with the bootstrap sample, and weights and saliences are recorded upon completion. Another bootstrap sample of 124 examples is drawn from the random sample, and the training is continued from the location of the final weights of the previous bootstrap sample. Again, weights and saliences are recorded at the completion of training for each repetitive bootstrap sampling. Upon the completion of B samples, the mean weights and saliences are calculated. Pruning decisions are determined from the analysis of these mean estimates.

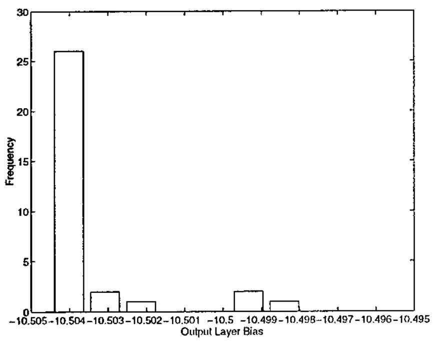


Figure 6. Bootstrap Distribution: Weight Initialized to Previous Result

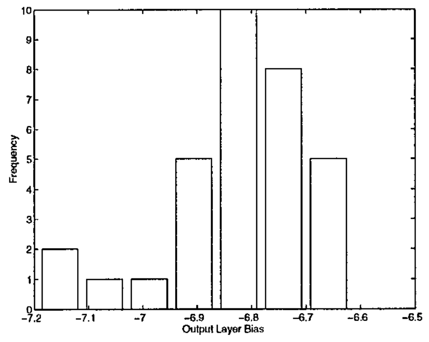


Figure 7. Bootstrap Distribution: Weight Initialized to 1st Result

Unfortunately, when subsequent bootstrap iterations are started from the previous bootstrap's final location to accelerate training, training continues in an improving location (near optimal state) such that successive iterations may require no training at all to achieve the error goal. The resulting parameter estimates remain unchanged, providing a biased result. Randomly selecting the initial weight set from previous bootstrap iterations also finds the most optimal weight set reappearing more frequently as the weights are recorded unchanged in the sample set. Both approaches remain practical with the aid of additional programming to force training via a flexible error goal. However, for purposes of this example, the algorithm is modified to always initialize subsequent bootstrap iterations with the weights recorded from the training of the first bootstrap sample. Successive iterations are less likely to succeed without training, thereby providing additional weight estimates. Figures 6 and 7 illustrate contrasting distributions of weight estimates from the bootstrap method using initial weights that hinder and foster retraining. Figure 3 illustrates a large sampling using the same process as that for Figure 7.

For naive simplicity, this pruning algorithm considers the magnitude of the mean weights for elimination with caution: weight magnitude pruning (27) in which the weights are ranked according to magnitude with the smallest deleted, can lead to improper pruning (23, 25). A small weight value may not be indicative of unimportance depending on the strength of the coupled output. According to Gorodkin (23), weight magnitude pruning is equivalent to the second derivative within the Taylor series expansion of the perturbation of error to be assumed a constant for all weights, when in fact, for the hidden to output weights, the second derivative is determined by the activation level of the hidden unit not necessarily the same as the rest. Likewise, for the input to hidden weights, the corresponding output weight figures in the derivative and they are typically not equal.

Although weight magnitude pruning is not the end goal of this dissertation, in this example, the weight magnitude method merely illustrates the iterative process and employment of the Tukey MCP. In the pruning analysis, the Tukey MCP uses the test statistic of Equation 7 to determine which weights are not significantly different from the smallest weight considered for network reduction. By zeroing these additional weights of no significant difference, pruning is accelerated.

The bootstrap technique recycles for additional pruning. By allowing the zeroed weights to retrain, training is accelerated (recall the advantage of larger networks). Also, the possibility exists that a once zeroed weight may gain significance as a result of a new found weight subject to elimination. An ever increasing number of weights are zeroed as

Table 1. Order of Weight Elimination

Cycle(s)	Weights Eliminated						
1	blue1	smile1	tie1	sqrhead2	tieless2	balloon3	hidbias1
2-5	flag3	octhead1	smile2	sqrhead2[a]	sword3	rndtorso1	sword2
6-11	balloon2	flag2	tieless1	rndtorso2	rndhead2	yellow1	sad1
12-16	sqrtorso2	balloon1	green1	sad2	tie2	hidbias3	sqrtorso 2
17-21	green2	tie3	sword2	yellow2	blue2	smile3	flag1
22-28	rndhead3	tieless3	sad3	hidbias2	octhead3	yellow3	blue3
29-31	red1	green3	octtorso2				

[a]Eliminated again in third cycle after regaining significance in second cycle

the pruning cycles to a stopping point. In this example, the neural network is reduced to a quarter its original number of connections.

3.4.2 Analysis of the Results. After 31 pruning cycles, the neural network connections reduces 75% from 58 to 14 weights. In seven of the cycles, the Tukey MCP finds an insignificant difference between the least weight (subject to pruning) and up to six other weights in the cycle. By zeroing those weights of no significant difference, thirteen cycles are avoided. In one instance, a once eliminated weight regarding head shape gains significance, only to be insignificant again in a subsequent pruning cycle. Table 1 shows the order in which weights are deleted.

In looking at Figure 8, note that the inputs of consequence relate directly to the Monks Problem 1 definition. Head and torso shape and the color red are the only inputs left to distinguish as to whether the robot answers the description of a robot with the same head as torso or wearing the color red. The results nearly mirror the network pruned by Hassibi's OBS (25), so the use of weights instead of saliences were not detrimental to the demonstration. Prior to zeroing the 44 weights to be pruned, the last training cycle produced zero error. Performed early in the computer code development, the algorithm had not retrained with the 44 weights *eliminated* which would be necessary to avoid large error (33%) by the remaining 14 weight estimates. Further work incorporates the retraining necessity. Also, additional pruning may produce a non-redundant input to hidden layer relationship, and possibly eliminate the second hidden node. This may occur if the network comprehends octagon as the absence of round and square. If the network starts with one hidden node, Setiono (51) shows that the Monks Problem 3 network could be pruned to six connections to describe the four distinguishing characteristics of the problem, allowing for one connection each from the bias and the hidden node.

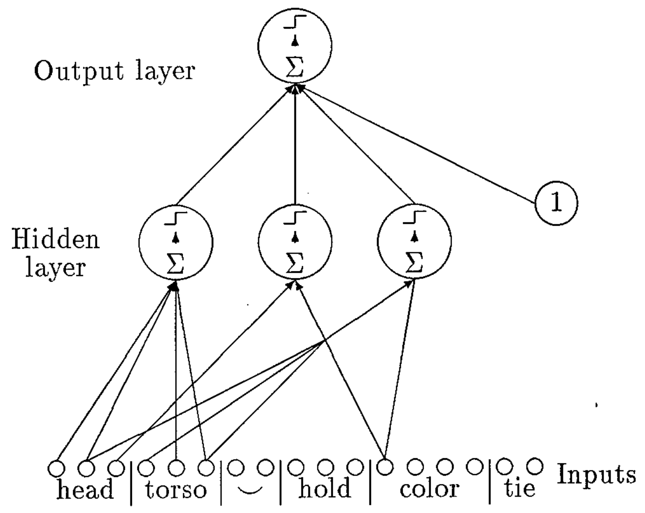


Figure 8. Monk's Problem 1 Neural Network Reduction by Weight Magnitudes using Tukey MCP

3.4.3 Critique of the Approach. In critique of this analysis, the Tukey MCP requires common variance and uncorrelated mean estimates of the parameters. In the case of the neural network machine, the mean weight estimates meet neither requirement. The heteroscedasticity and correlation within the estimates lead to a larger MSE in Equation 7. This produces a more conservative test that promotes a greater likelihood of no significant difference between the weight subject to pruning and weights near valued. The test eliminates near valued weights more readily. In the first cycle when seven weights are pruned, if the means are ordered and the subsequent six are compared with the least weight, a t-test finds only three of the six not significantly different from the least weight. The Newman-Keuls test produces a result similar to the Tukey MCP and Duncan's new multiple range test comes in between the t-test and the others (24).

Gibbons et al. (20) and Hsu (29) consider subset selection procedures for homoscedastic, uncorrelated sample means as a conservative analogue to the ordered t-test approach. Subset selection considers a nonempty set of populations with a probability of a correct selection of the subset containing the best population. Given the maximum (or minimum) sample mean, a one-sided confidence interval is drawn to exclude other sample means as the true extremum. Those samples within the interval may be considered of no significant difference. This approach supports multiple comparison pruning in that the sample means can be ordered and the smallest salience compared to others within a confidence interval. However, neither homoscedasticity nor independence may be assumed.

Regarding nonequal variances, the Behrens-Fisher problem addresses heteroscedasticity for independent samples. Sachs (47) describes the approximating routine for Behrens-Fisher difference of two means of possible unequal variance as a modified t-test. Stein (52) looks to a two-stage procedure devoid of variance in the power function. Acknowledging the nonexistence of single-sample selection procedure independent of variance, Dudewicz et al. (14, 15, 16) furthers the selection process with the Heteroscedastic Method for ranking and selection with unknown and unequal variances.

In order to account for the correlation between sample means, procedures of multiple range tests, confidence intervals, and pairwise comparisons for dependent means are developed by Kramer (31), Dunn (17), and Brown (11). Covariates join the studentized range distribution as an integral part of the decision rule. Thus, selecting the subset with the best population, that is the least weight or salience, is possible for heteroscedastic, correlated components. The resulting subset is that to be pruned in the algorithm.

Correlated means can also be compared using nonparametric techniques. The rank sum statistic of each parameter can be used for pairwise comparisons in a single-step test procedure (28). From the Friedman Test (19), Sachs (52) considers approximate multiple comparisons along the approach of Student, Newman and Keuls, and Wilcoxon and Wilcox. The distribution-free comparison of correlated samples bypasses the need for covariates and becomes an alternate approach for the pruning decision rule. Needless to say, the introductory example of this section illustrated some simple algorithmic applications. However, in accounting for the correlated means and departing from the weight magnitude method, Section 3.5 details the Kramer extension of the Tukey MCP, the performance on salience measures, and the appropriateness of this method.

3.5 Tukey-Kramer Multiple Comparison Pruning

In the previous section, the Tukey MCP demonstrated the ability to prune subsets of weights within a single decision while clarifying the need to account for the correlation between sample means. The procedures of multiple range tests, confidence intervals, and pairwise comparisons for dependent means as developed by Kramer (31), Dunn (17), and Brown (11), involve the covariates as an integral part of the studentized range decision rule. The estimated variances, $c_{ii}s^2$, and covariances, $c_{ij}s^2$, establish a standard error term for the difference of two means by $Z_{ij}^2 = \left(c_{ii}s^2 - 2c_{ij}s^2 + c_{jj}s^2\right)/2$. The simultaneous confidence intervals,

$$\mu_i - \mu_j \in \left[\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j \pm q(1-\alpha; P; v)Z_{ij}\right] \quad (1 \le i \le j \le P)\,, \tag{8}$$

have confidence level at least $1 - \alpha$. Selecting the subset with the least salience is possible for heteroscedastic, correlated components. The resulting subset of saliences identify the weights to be eliminated. This approach is called the Tukey-Kramer MCP. Using the same Monks Problem as in Section 3.4.1, the Tukey-Kramer MCP performance is demonstrated on the saliences of the weights.

OBS is again used as the comparative pruning method. In the pruning analysis, the Tukey-Kramer MCP determines which mean saliences are not significantly different from the smallest mean salience considered for network reduction. By zeroing additional weights of no significant difference in salience, pruning is accelerated. If more than one weight is removed, the reduced network is retrained, otherwise the weight update procedure of OBS is employed for additional analysis of the mean saliences until a stopping criterion is reached.

3.5.1 Multiple Salience Comparison Results. After 18 pruning cycles of MCP, the number of neural network connections reduced to a fourth, from 58 to 15 weights. In six of the cycles, the Tukey-Kramer MCP found an insignificant difference between the least mean salience (weight subject to pruning) and up to seven other saliences in the cycle at $\alpha = 0.05$. By zeroing the associated weights with saliences of no significant difference, twenty-five cycles were avoided.

In looking at Figure 9, note that the remaining inputs of consequence relate directly to the Monks Problem 1 definition. Head and torso shape and the color red are the only inputs left to distinguish as to whether the robot answers the description of a robot with the same head as torso or wearing the color red. The results nearly mirror the network pruned by Hassibi's OBS (25). Final network retraining has been added to the algorithms for the rest of the dissertation. Retraining the algorithm with the 43 weights *eliminated* produces an error of 8.33%, similar to OBS.

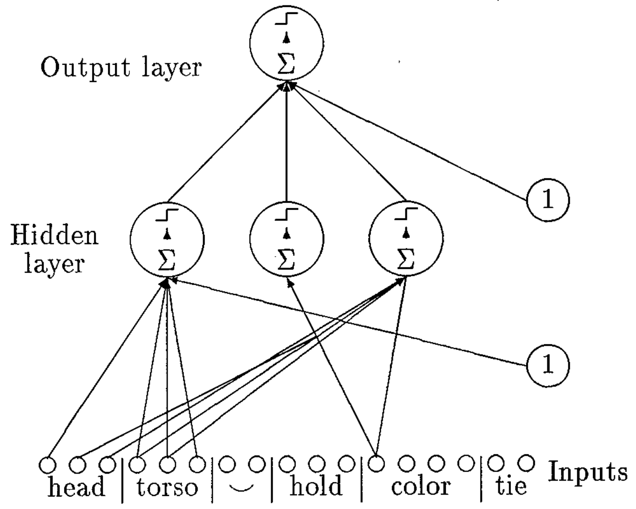


Figure 9. Monk's Problem 1 Neural Network Reduction by Salience Magnitudes Using Tukey-Kramer MCP

While the Tukey-Kramer MCP was as successful as OBS in trimming the network, the computational time required to collect enough bootstrap resamples for mean weight estimation was much larger for MCP than for OBS. The floating point operations (flops) for MCP were 40.1 giga-flops compared to 2.75 giga-flops for OBS. This order of magnitude difference in part may be attributed to the relatively small size of the initial network. In that only 43 weights were ultimately eliminated, the MCP was hard pressed to repeatedly eliminate large numbers of weights to justify the resampling.

To demonstrate the capability of MCP in a favorable arrangement, consider an initial network of ten hidden nodes. With 191 weights in the network, the training set must be expanded from 124 exemplars to something greater than the number of weights. In this case 248 exemplars were drawn for the training set. After resampled training, the

Table 2. Initial MCP Weight Elimination

Inputs	Hidden	Weights	Ave Pruned
17	3	58	5.45
17	5	96	7.79
17	10	191	60.74

MCP performed a single subset elimination on those weights whose saliences were not significantly different from the least salience at $\alpha = .05$. In 119 trials, MCP eliminated 30 to 90 weights, or approximately 61 weights on average. In competition, OBS was permitted to eliminate the same number of weights for each of the 119 trials. In this situation, MCP equaled OBS accuracy in all but two of the trials. Meanwhile, MCP required 13% less flops than OBS. MCP averaged 113 giga-flops compared to 131 giga-flops for OBS. The paired t-test found the reduction in computational burden significant with $p < 0.0001$.

So, with larger networks, the size of the subset eliminated by MCP grows as evident by the trend in Table 2. Since OBS requires an additional iteration for each weight MCP subset eliminates, the difference in floating point operations favors MCP to surpass OBS in efficiency.

The Tukey-Kramer MCP limitation is its need for mean estimates to satisfy the hypothesis test. The computational burden of bootstrap iterations ceases by developing a single sample test. The work of Pareto (42), though focused on income distributions, is often cited for the applicability of his distribution to logarithmic models. By applying the notion of trivial many and vital few to the saliences of the network weights, Pareto distribution theory replaces bootstrap resampling with single sample analysis. Section 3.6 develops the single sample multiple comparison approach to compete with single elimination methods of pruning algorithms.

3.6 Pareto Pruning

In contrast to Tukey-Kramer MCP's computational burden of mean estimation and comparison, Pareto pruning considers the distribution of a single sample of saliences. The classical Pareto (42) distribution is modeled by

$$\mathcal{F}(x) = Cx^{-\mathrm{a}} \qquad x \geq 0,$$

determining the proportion of individuals in a population with income exceeding x. Pareto used the distribution to assert an underlying law for the distribution of income: that the Pareto parameter, a, was invariant about the value 1.5 under changes in population. The development of similar income models continued in the early 1900's, but the distribution took on new significance in the 1950's when J.M. Juran popularized the Pareto diagram as means of quality control. Rather than model the wealthy few in a population, Juran focused the use of the distribution on the vital few processes that hindered quality. In a similar fashion for neural networks, the Pareto diagram in Figure 10 illustrates the vital few saliences that determine the necessary weights and the trivial many saliences in the tail of the distribution.

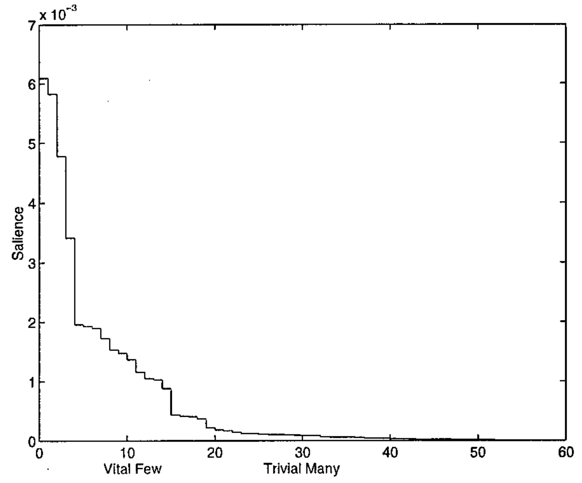


Figure 10. Pareto Diagram of Network Saliences

From a single sample of saliences, estimates of the distribution parameters can be used to make comparisons and subset elimination similar to the Tukey-Kramer MCP without the computational burden of bootstrapping.

Arnold (5) provides a substantial review of the Pareto distribution. The two-parameter form of the Pareto distribution employs the Pareto or shape parameter, a, and the location or cut-off value, k, in the probability density function

$$\mathrm{f}(x) = \begin{cases} 0 & x < \mathrm{k} \\ \mathrm{ak^a}/x^{\mathrm{a}+1} & x \geq \mathrm{k} \end{cases} \tag{9}$$

and from Equation 9, the resulting cumulative distribution function

$$\mathcal{P}(x) = 1 - \left(\frac{\mathrm{k}}{x}\right)^{\mathrm{a}}, \quad 0 < \mathrm{k} \leq x, \quad \mathrm{a} > 0. \tag{10}$$

The x in the Pareto distribution is the salience measure of each weight for this dissertation.

Quandt's (43) maximum likelihood estimators (MLE) of a and k, comparable to Muniruzzaman's (39), are

$$\hat{\mathrm{k}} = \min_{1 \leq i \leq P} x_i, \tag{11}$$

and

$$\hat{\mathrm{a}} = \frac{P}{\sum_{i=1}^{P} \log(x_i/\hat{\mathrm{k}})}. \tag{12}$$

The unbiased estimators are given by

$$\tilde{\mathrm{a}} = \left(\frac{P-2}{P}\right)\hat{\mathrm{a}}, \quad P > 2, \tag{13}$$

and

$$\tilde{\mathrm{k}} = \left[1 - \frac{1}{(P-1)\hat{\mathrm{a}}}\right]\hat{\mathrm{k}} \tag{14}$$

as derived by Saksena (48, 49) using Malik's (36) results on the distributions and independence of $\hat{\mathrm{k}}$ and $\hat{\mathrm{a}}$. Arnold (5) summarizes that the unbiased estimates of Equations 13 and 14 improvement in behavior for fixed finite value P also benefit from a generalized MSE uniformly smaller than the MLE of Equations 11 and 12. Baxter (8) and Saksena et al. (49) independently show that the minimum variance unbiased estimates

(MVUE) $\tilde{\mathrm{k}}$ and $\tilde{\mathrm{a}}$ are asymptotically efficient and the relation of variances for MVUE versus MLE are as follows:

$$Var(\tilde{\mathrm{a}}) = \frac{\mathrm{a}^2}{P-3} < Var(\hat{\mathrm{a}}) = \frac{P^2\mathrm{a}^2}{(P-2)^2(P-3)} \quad \text{if} \quad P > 3,$$

and

$$Var(\tilde{\mathrm{k}}) = \frac{\mathrm{k}^2}{\mathrm{a}(P-1)(P\mathrm{a}-2)} < Var(\hat{\mathrm{k}}) = \frac{P\mathrm{ak}^2}{(P\mathrm{a}-1)^2(P\mathrm{a}-2)}$$

if $\mathrm{a} < 2$ and $P > \frac{1}{\mathrm{a}(2-\mathrm{a})}$.

Kang and Cho (30) find that if one of the parameters, a or k, is known, the biased jackknife estimate of the other parameter compares with the MLE. They further derive a biased minimum risk estimator (MRE) that has a smaller MSE than both the MLE and the MVUE. They then propose a MRE for the case of both parameters unknown; however, the MRE still retains a bias for a minimal improvement of MSE. The use of other estimates of a and k may be worthy of future work.

The implementation of Pareto Pruning is a modification of existing pruning algorithms, where the Pareto distribution supports the decision to remove multiple connections from the network. For algorithmic simplicity, the unbiased estimate of the cut-off value k is used in a confidence interval similar to the Tukey-Kramer Confidence Interval of Equation 8. The confidence intervals are constructed for

$$\mu_i - \mu_j \in \left[\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j \pm (1-\mathrm{CDF})^{-\frac{1}{\tilde{\mathrm{a}}}} \cdot \tilde{\mathrm{k}}\right] \quad (1 \le i \le j \le P)$$

where CDF is the desired level of pruning based on the cumulative distribution function of Equation 10. Based on the comparable level of pruning by Tukey-Kramer MCP, 20% elimination is a desirable level for the CDF term. As the distribution function rises to the 20% level, the steep decline on the left side of the Pareto density function encompasses only the trivial saliences. Essentially, saliences less than $(1-\mathrm{CDF})^{-\frac{1}{\tilde{\mathrm{a}}}} \cdot \tilde{\mathrm{k}}$ are grouped in the trivial many subset for elimination. Weight updates are based on the optimal weight change prescribed by the largest salience within the subset eliminated.

3.6.1 Pareto Pruning versus Tukey-Kramer MCP. To compare the performance of the Tukey-Kramer MCP with Pareto Pruning, consider Monk's Problem 1 for a network beginning with 58 connections between the input layer, three hidden nodes, and one output node. Minimally, the pruning algorithm is expected to detect which inputs are important to discern the desirable characteristics. The pruning algorithm may test the reduced network with the entire 432 legal examples.

For both methods, a random sample of 124 robots is drawn from the 432 legal examples. The network is trained with the sample, and weights and saliences are recorded upon completion. Pruning decisions are determined from the analysis of these estimates. In the pruning analysis, Pareto pruning determines which saliences are near the least salience considered for network reduction. The histogram of Figure 11 illustrates the dense collection of trivial weights in a Pareto-like distribution.

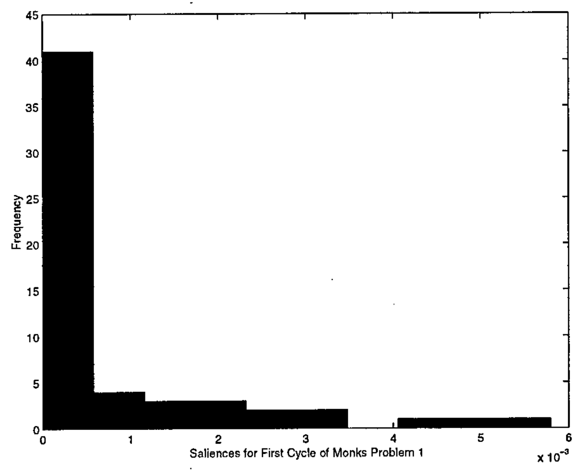


Figure 11. Network Salience Distribution

By zeroing additional weights with trivial salience within a CDF level, pruning is accelerated. The weight update procedure of OBS is employed based on the largest salience eliminated in that iteration, propagating an updated Hessian and analysis of the remaining saliences until a stopping criterion is reached. If the stopping criterion is exceeded, the last iteration is undone and OBS is employed to remove additional weights individually until the stopping criterion is again exceeded. Then the last iteration is undone and pruning is complete.

The Tukey-Kramer MCP bootstrap resamples as before to accomplish the multiple comparisons of saliences and subset elimination of weights whose saliences are not significantly different. Retraining occurs when more than one weight is eliminated; otherwise, the same weight update procedure is applied to the bootstrap estimates for another iteration of MCP. If the stopping criterion is exceeded, the last iteration is undone and pruning is complete.

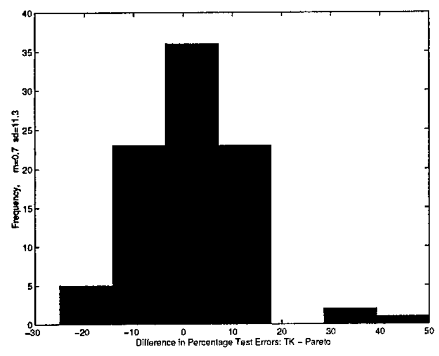


Figure 12. Difference in Percentage Test Error for Tukey-Kramer vs Pareto Pruning

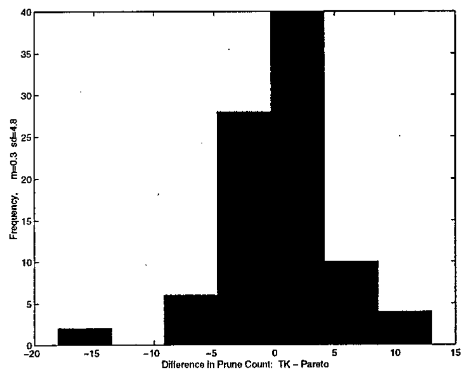


Figure 13. Difference in Prune Count for Tukey-Kramer vs Pareto Pruning

One hundred runs of network reduction of Monk's Problem 1 compare the performance, degree of pruning, and computational burden of Tukey-Kramer MCP and Pareto Pruning. In Figures 12 and 13, the difference in performance and degree of pruning is found to be not significantly different from zero using the Bonferroni t-test with $\alpha = 0.05$ and $g = 3$ combinations. However, the paired t-test finds the floating point operations of Tukey-Kramer MCP to be significantly greater ($p < 0.0001$) than those for Pareto Pruning as seen in Figure 14, similar to the comparison between Tukey-Kramer MCP and OBS. Whereas Tukey-Kramer MCP is too computationally intensive to compete with Pareto Pruning and OBS in efficiency, Pareto Pruning operates on the same order of magnitude of floating point operations as OBS and thus warrants further comparison.

3.6.2 Pareto versus OBS Pruning. The implementation of Pareto pruning is a modification of existing pruning algorithms, where Pareto analysis supports the decision to remove multiple connections from the network. Monk's Problems are used to demon-

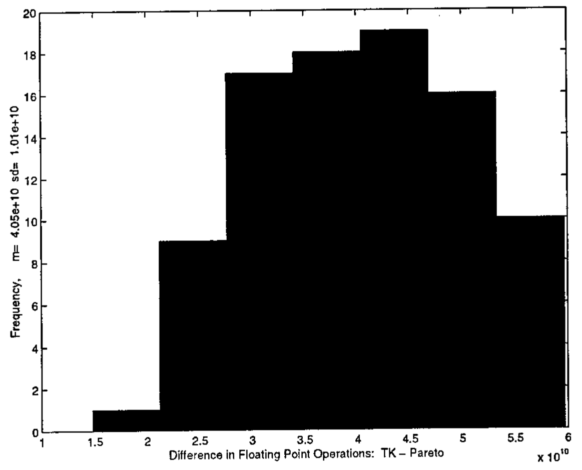


Figure 14. Difference in Floating Point Operations for Tukey-Kramer vs Pareto Pruning

strate the performance of pruning algorithms like OBS or its competitors. Likewise, the performance with and without Pareto analysis can be demonstrated by example.

In this section, OBS is used as the comparative pruning method. The network of Figure 5 is trained for Monk's Problem 1 to discern if the robot's head shape and body shape are the same, or if the coat color is red. The training and test errors are zero or minimal for the complete network structure. As before, the errors increase as the network is pruned to sparse structure of relevant inputs and necessary hidden units. However, as Figure 15 indicates, the Bonferroni t-test ($\alpha = 0.05$ and $g = 3$ combinations) finds no significant difference in error when comparing OBS with Pareto pruning. The numbers of connections pruned are also not significantly different at $\alpha = 0.05$ as shown in Figure 16. In contrast, from the paired t-test, the amount of floating point operations required to reach the final reduced network favors Pareto pruning significantly ($p < 0.0001$) as shown in Figure 17. The decision to remove more than one trivial connection at a time as a function of the estimates of the Pareto CDF reduces the number of iterations and Hessian inversions to reach the stopping criterion.

The example can be broadened as in Monk's Problem 2 by training the neural network to discern if two of the six features of the robots have the first value. All features play a role in the network training. Nonetheless, as Figures 18 and 19 indicate, the Bonferroni t-test ($\alpha = 0.05$ and $g = 3$ combinations) finds no significant difference in performance nor

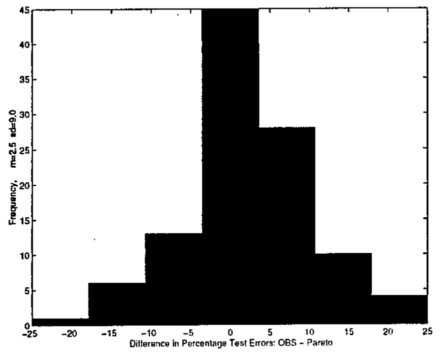


Figure 15. Monk's 1, Difference in Percentage Test Error for OBS vs Pareto Pruning

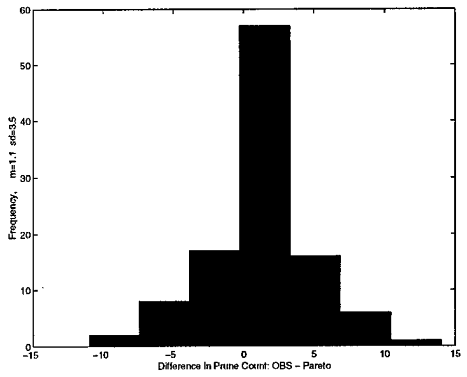


Figure 16. Monk's 1, Difference in Prune Count for OBS vs Pareto Pruning

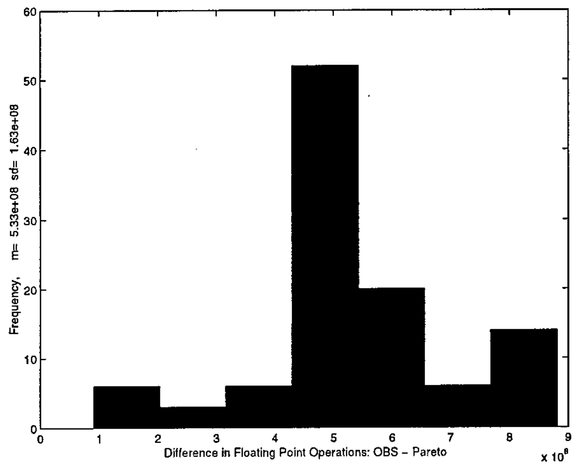


Figure 17. Monk's 1, Difference in Floating Point Operations for OBS vs Pareto Pruning

level of pruning for OBS as compared with Pareto pruning. Figure 20 and the paired t-test indicate that Pareto pruning has significantly reduced ($p < 0.0001$) the computational burden with fewer iterations to reach the stopping criterion.

Monk's Problem 3 dictates training the neural network to discern if the robot's holding a sword and its coat color is red, or if the coat color isn't blue and body shape isn't octagon, with a corrupted training set. The 5% corruption of training set actually improves the network's ability to generalize with lower test errors, but the error difference between using Pareto or not in Figure 21 is not significant at $\alpha = 0.05$ with $g = 3$ combinations. Nor is the final complexity, with or without Pareto, significantly different

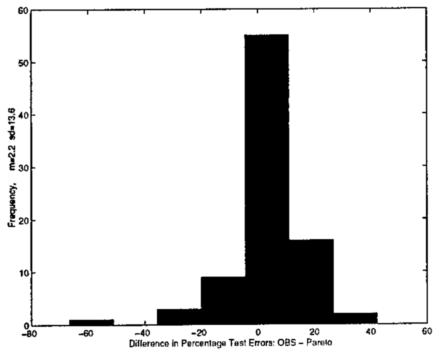


Figure 18. Monk's 2, Difference in Percentage Test Error for OBS vs Pareto Pruning

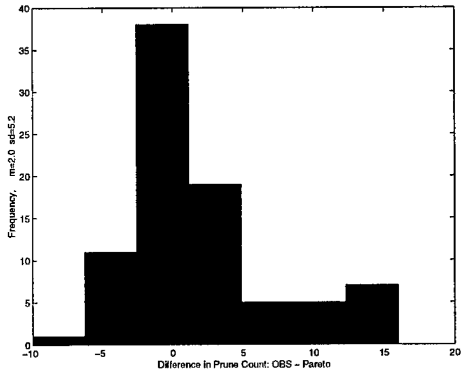


Figure 19. Monk's 2, Difference in Prune Count for OBS vs Pareto Pruning

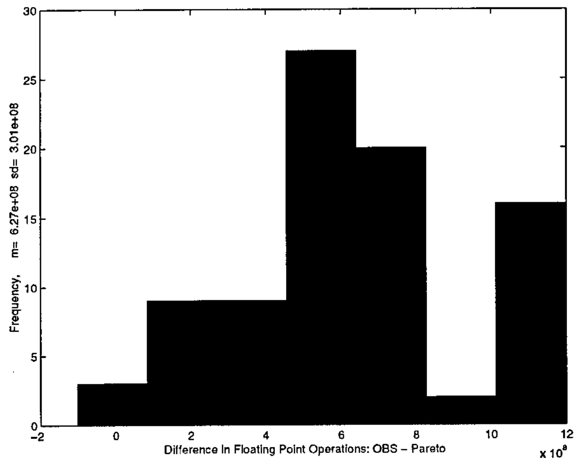


Figure 20. Monk's 2, Difference in Floating Point Operations for OBS vs Pareto Pruning

at $\alpha = 0.05$ when the stopping criterion is reached, as shown in Figure 22. For Monk's Problem 3, the paired t-test finds Pareto pruning significantly less computationally costly ($p < 0.0001$) than OBS, as evident in Figure 23.

Even though Pareto pruning proves to be an effective and efficient pruning approach for all three of the Monk's Problems involving the 17 input binary system, the Pareto-like distribution of Figure 11 is rejected by a liberal Chi-Square goodness-of-fit test (40) with a $p-$value < 0.005. While Pareto Pruning demonstrates the ability to subset eliminate weights just like Tukey-Kramer MCP (but with one sample), the use of the Pareto parameters' estimator is not justifiable. Although the distribution of the saliences suggest

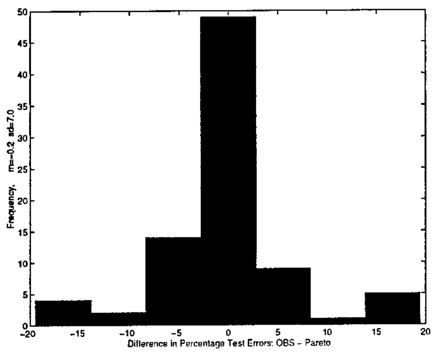


Figure 21. Monk's 3, Difference in Percentage Test Error for OBS vs Pareto Pruning

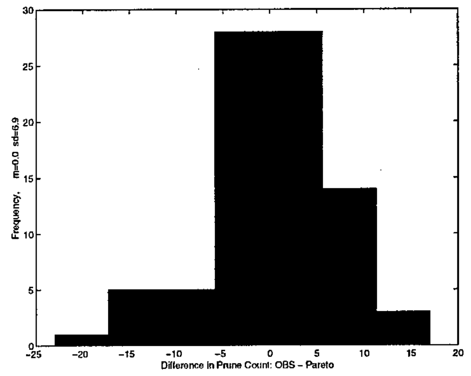


Figure 22. Monk's 3, Difference in Prune Count for OBS vs Pareto Pruning

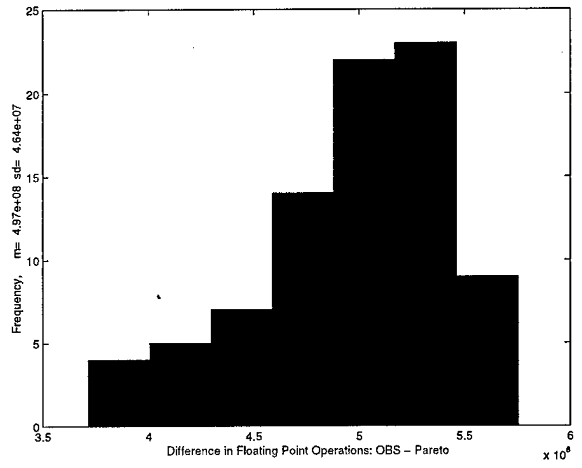


Figure 23. Monk's 3, Difference in Floating Point Operations for OBS vs Pareto Pruning

a disbursement of a trivial many against a vital few, the underlying distribution is found to behave more like a Weibull distribution. Section 3.7 considers the characteristics of the Weibull distribution and its fit to the distribution of saliences.

3.7 Weibull Pruning

Although the neural network saliences appear to have a Pareto distribution with the precipitous decline from the left, the goodness-of-fit tests suggest that the drop is better described by a Weibull distribution. Developed in 1937, W. Weibull's distribution is primarily considered for life data and failure forecasting. Abernethy (1) provides a substantial review of the Weibull distribution. The two-parameter form of the Weibull distribution employs the slope or shape parameter, β, and the scale or characteristic life parameter, ζ, in the probability density function

$$\mathrm{f}(x) = \frac{\beta}{\zeta}\left(\frac{x}{\zeta}\right)^{\beta-1} e^{-(x/\zeta)^{\beta}} \qquad x \geq 0 \tag{15}$$

and the resulting cumulative distribution function

$$\mathcal{F}(x) = 1 - e^{-(x/\zeta)^{\beta}} \qquad x \geq 0. \tag{16}$$

The x in the Weibull distribution is the salience measure of each weight for this dissertation. The MLE of β, denoted $\hat{\beta}$, satisfies

$$\frac{\sum_{i=1}^{P} x_i^{\hat{\beta}} \log x_i}{\sum_{i=1}^{P} x_i^{\hat{\beta}}} - \frac{1}{P}\sum_{i=1}^{P} \log x_i - \frac{1}{\hat{\beta}} = 0, \tag{17}$$

and dependent on $\hat{\beta}$, the MLE of ζ is

$$\hat{\zeta} = \left(\frac{\sum_{i=1}^{P} x_i^{\hat{\beta}}}{P} \right)^{\frac{1}{\hat{\beta}}}. \tag{18}$$

The Weibull distribution uses the data to select the distribution and fit the parameters (1). Figure 24 illustrates the flexibility of Equation 15 to encompass a variety of data representations. For β less than one, f has a precipitous decline on the left like a Pareto distribution. When $\beta = 1$, the Weibull is the Exponential distribution. For β greater than one, the Weibull distribution appears more like a lognormal distribution with a starting point at the origin.

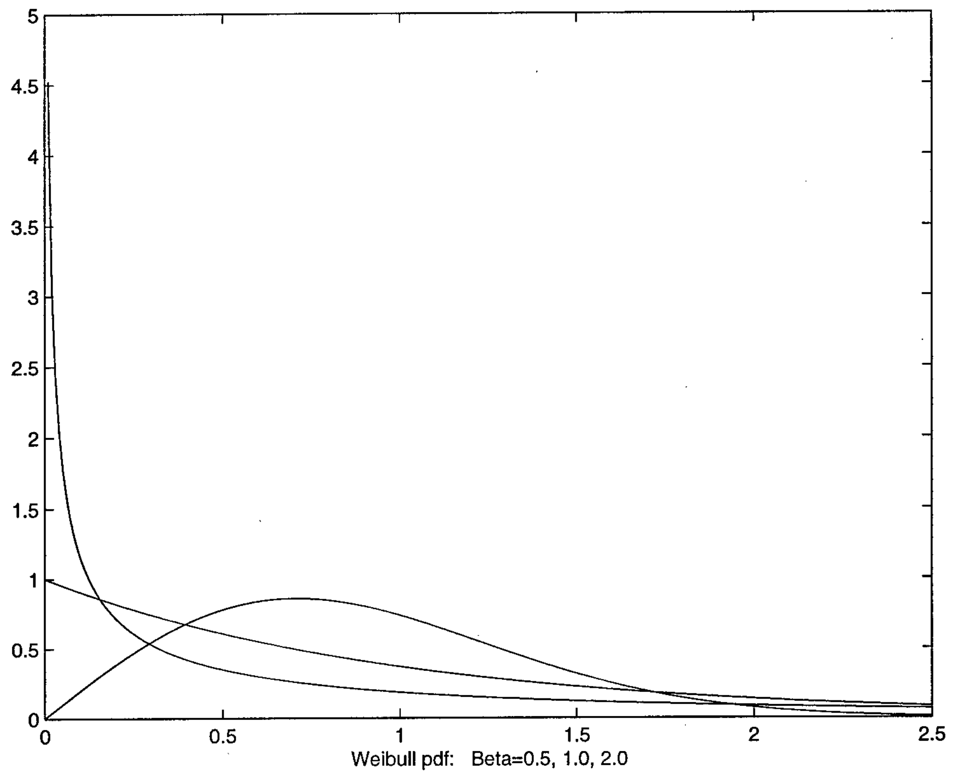


Figure 24. Weibull Probability Density Function for Various Slope Parameter

The implementation of Weibull Pruning, like Pareto Pruning, is a modification of existing pruning algorithms, where the Weibull distribution supports the decision to remove multiple connections from the network. A goodness-of-fit test is part of the algorithm decision prior to pruning a subset of weights. If the Weibull distribution is rejected, the algorithm performs a single weight elimination. Likewise, the notion of *trivial many saliences* dictates subset pruning only when β is less than one, see Figure 24. For algorithmic simplicity, the MLEs of Equations 17 and 18 are used in a confidence interval similar to the Tukey-Kramer Confidence Interval of Equation 8. The confidence intervals are constructed for

$$\mu_i - \mu_j \in \left[\bar{\mathrm{L}}_i - \bar{\mathrm{L}}_j \pm \hat{\zeta} \cdot \left(\log \frac{1}{1 - \mathrm{CDF}} \right)^{\frac{1}{\hat{\beta}}} \right] \quad (1 \le i \le j \le P) \tag{19}$$

where CDF is the desired level of pruning based on the cumulative distribution function of Equation 16. Based on the comparable level of pruning by Tukey-Kramer MCP and the results of Pareto Pruning, 20% elimination is a desirable level for the CDF term. As the distribution function rises to the 20% level, the steep decline on the left side of the Weibull density function encompasses only the trivial saliences. A less aggressive 10% elimination, although not as efficient, may be appropriate if the β is closer to unity, when the Weibull distribution becomes the Exponential distribution. In either case, saliences less than $\hat{\zeta} \cdot \left(\log \frac{1}{1-\text{CDF}}\right)^{\frac{1}{\beta}}$ are grouped in the trivial many subset for elimination. Weight updates are based on the optimal weight change prescribed by the largest salience within the subset eliminated.

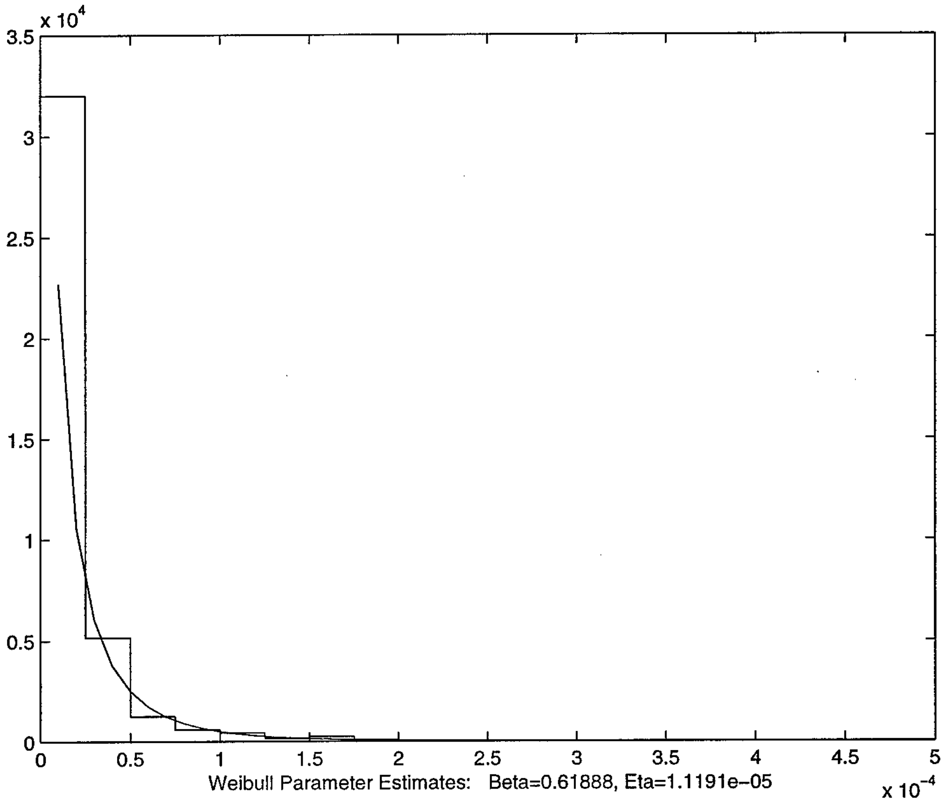


Figure 25. Histogram of 1001 Saliences and the Weibull Fit

The goodness-of-fit of the neural network saliences tests favorably for the Weibull distribution. For a 48 input, 20 hidden node, single output network with 1001 weights, the saliences fit a Weibull distribution with a p-value > 0.25 as shown in Figure 25. The steep decline caused by $\hat{\beta} = 0.62$ allows for a satisfactory grouping of trivial saliences.

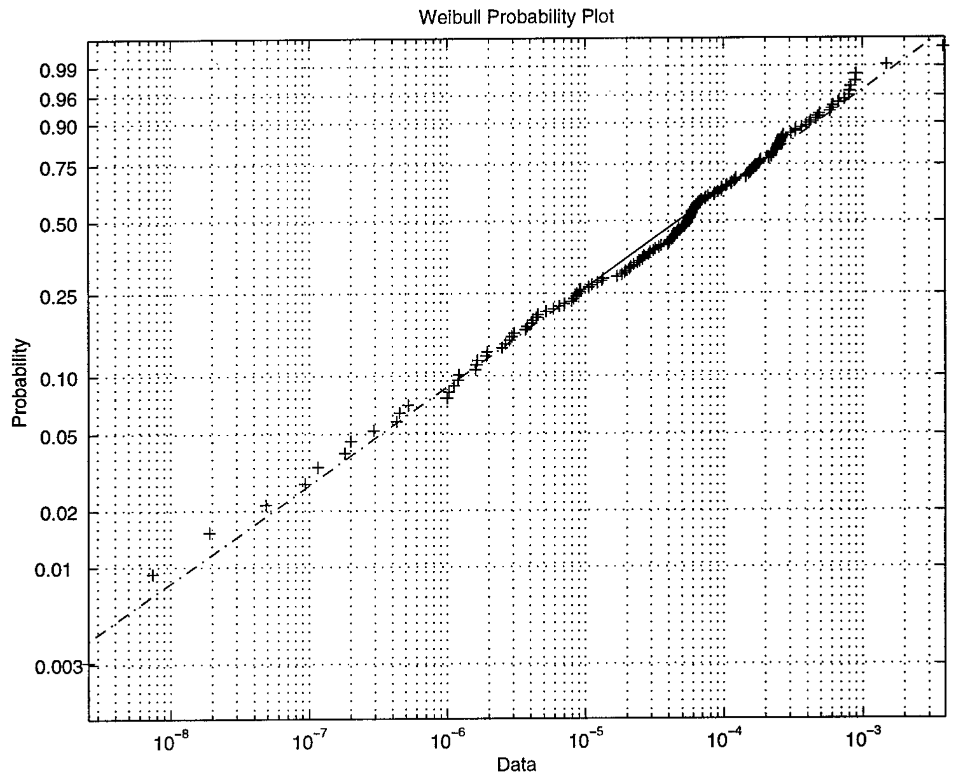


Figure 26. Weibull Probability Plot for Saliences of a Neural Network

The linear fit of the probability plot for a smaller 16 input, 9 hidden node network in Figure 26 supports the acceptance of the Weibull distribution by the Anderson-Darling test at a level beyond 0.25. Provided $\hat{\beta}$ is less than one, Weibull Pruning is an appropriate alternative to the Tukey-Kramer MCP. In Chapter IV, the efficacy of Weibull Pruning and Tukey-Kramer MCP is tested in an experiment for statistical measure.

IV. Experiments

The Monk's Problems provide a binary system of inputs for training and pruning neural networks. In Chapter III, the Monk's Problems finds the performance of Tukey-Kramer MCP and Pareto Pruning in comparison with OBS is not significantly different at $\alpha = 0.05$. Meanwhile the efficiency of the two straddle OBS: Pareto pruning is more efficient, Tukey-Kramer MCP is less. However, in this experiment, Weibull distribution theory replaces the Pareto distribution for the single sample distribution estimates determining the subset elimination. A goodness-of-fit is included in Weibull Pruning prior to acting upon a subset of the saliences. In order to better generalize the characteristics between the methods developed in this dissertation and an established pruning method, a simulation experiment is proposed using factorial design procedures.

4.1 Design of Experiment

The number of inputs and hidden nodes to the single output response determines the overall network structure. For this simulation, the input vector lengths or number of inputs are 4 and 16. The hidden nodes vary as 3, 6, and 9. The combination of the two network attributes produce 5 different connection counts of 19, 37, 55, 109, and 163. The input vector data, $\mathbf{x}$, are assumed to be normally distributed with mean, $\mathbf{0}$, and variance-covariance matrix, $\mathbf{I}$ (dimensions supportive of the input structure of the network). The decision of the underlying network complexity to obtain the training set outputs and the size of the training set comes from a previous experiment. The training set size is 256. The corresponding outputs are calculated from a network instilled with a reduced complexity. The reduced internal complexity is the percentage of connections initialized to zero, 25% for this simulation. Only the biases and weights connecting the inputs and hidden nodes are randomly chosen for an initial value of zero. The rest of the weights have a generated value assumed to be normally distributed with mean, $\mathbf{0}$, and variance-covariance matrix, $\mathbf{I}$. Given $\mathbf{x}$ and $\mathbf{w}$ (the weights), the corresponding outputs are calculated to complete the training set. A test set is also produced in the same manner.

Before incorporating the Tukey-Kramer MCP and OBS procedures in the simulation, Weibull Pruning is considered at both the 0.1 and 0.2 CDF level to measure the aggressiveness of pruning within the range of Tukey-Kramer MCP. Given a simulation data set, the networks train to an error goal and Weibull Pruning proceeds independently at the two CDF levels. To accelerate data collection, the computer simulation is run on separate machines for each network architecture. For the six different network structures, ten replications provide responses of the prune count, floating point operations, initial MSE, training MSE, and test MSE for both CDF levels. The prune count is the number of weights eliminated from the network and measures the depth at which the pruning approach is able to reduce the network complexity. The number of floating point operations is a measure of the computational burden of the algorithm; or conversely, the efficiency. The MSEs are the mean square error of the network's output response prior to pruning to assure a common start, and after pruning on both the training and test data set to measure the network's ability to remain trainable and predictable. The Weibull results determine which CDF level of Weibull Pruning will be used in comparison to the OBS and Tukey-Kramer MCP.

With single Weibull CDF level and the other two pruning approaches being considered, the simulation is a $2 * 3^2$ factorial design as shown in Table 3. For each treatment combination, ten replications provide responses of the prune count, floating point operations, initial MSE, training MSE, and test MSE for each pruning approach.

4.2 Computational Resources

Matlab's Neural Network Toolbox (13) is the computer program used to perform the simulation. Matlab's trainbpx function trains a feed-forward network with fast backpropagation. The speed of the backpropagation is enhanced by the use of momentum to find better solutions and adaptive learning rates to shorten training time. Momentum allows the network to respond to both the local gradient and previous trends in a fractional apportionment. The momentum of previous trends allows the network to roll through the local minimums that gradient descent would be content to rest. The resulting weight change is a convex combination where the momentum constant apportions a fraction of

Table 3. Simulation Factorial Design

Run	Hidden Units	Inputs	Method
1	3	4	OBS
2	3	4	Weibull
3	3	4	T-K MCP
4	3	16	OBS
5	3	16	Weibull
6	3	16	T-K MCP
7	6	4	OBS
8	6	4	Weibull
9	6	4	T-K MCP
10	6	16	OBS
11	6	16	Weibull
12	6	16	T-K MCP
13	9	4	OBS
14	9	4	Weibull
15	9	4	T-K MCP
16	9	16	OBS
17	9	16	Weibull
18	9	16	T-K MCP

the previous weight change with a fraction of the gradient descent weight change. If the new error exceeds the old error by 4%, the new weights are discarded and the learning rate, η, is modified by a multiplier of 0.7. Otherwise, the new weights are kept, and if the new error is less than the old error, η is increased 5%. The learning rate is increased during stable learning, and decreased when large error increases occur until stable learning resumes. Trainbpx uses batch training where the entire training set is fed through prior to weight changes. The function is designed for quick learning and avoiding local error minimums.

Matlab's Statistical Toolbox provides various functions for estimating Weibull parameters, inverting distribution functions, and providing probability plots. The toolbox provides various random number generators to create the data sets. Other functions within the toolbox overlap the greater capabilities of statistical software packages JMP (34) and ExpertFit (32) used in the analysis.

4.3 Experimental Results

The following sections discuss the various contrasts and comparisons between the pruning methods in an attempt to reveal their appropriateness.

4.3.1 Weibull Pruning Levels. Weibull Pruning checks for the goodness-of-fit of the saliences to a Weibull distribution and for the MLE of the shape parameter, β, to be less than one before grouping the trivial many saliences in the steep decline of the left side of the distribution. The level of Weibull Pruning is intended to encompass the weights having salience not significantly different from the least salience that Tukey-Kramer MCP would prune using multiple comparison procedures. In contrast, Weibull Pruning uses one sample distribution of saliences, while Tukey-Kramer MCP requires B bootstrap samples to make the pruning decision. However, Tukey-Kramer MCP uses statistical inference to determine which saliences are no different than the least salience whose weight is subject to pruning. The decision to prune any of those saliences is equally valid and their corresponding weights are eliminated as a group. Weibull Pruning acknowledges the presence of a subset in the left side of the distribution but removes the weights with less insight. The level by which Weibull Pruning eliminates the weights may have an affect on the overall performance of the approach. To compare performance, the simulation performs Weibull Pruning at both the 10% and 20% level.

For the 10 replications of each network structure, responses are recorded upon the completion of each CDF level of Weibull Pruning. The independent machines merge the response data to a single file for analysis. The responses from any given machine can be time sequenced, but not the total 120 values for any given response. Figure 27 presents a typical random pattern found for the ANOVA residuals from the simulation experiment. In this figure, the final test MSE is the response. The Durbin-Watson $p-$value $= 0.1967$ which suggests an independence of replications within the treatment.

Upon checking the assumption of normality, a typical residual pattern found is shown in Figure 28. In this figure, the final test MSE is the response. Although the Shapiro-Wilk W test rejects the null hypothesis for normality, the data has a symmetric normal-like distribution about zero. The studentized residuals display an appropriate number of

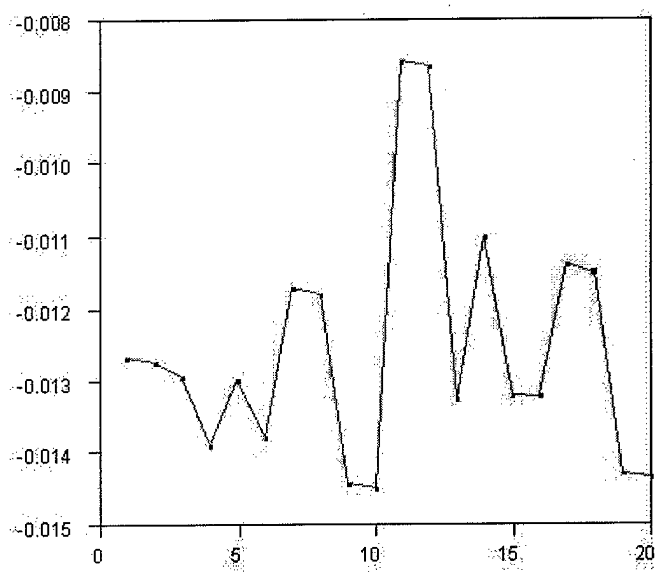
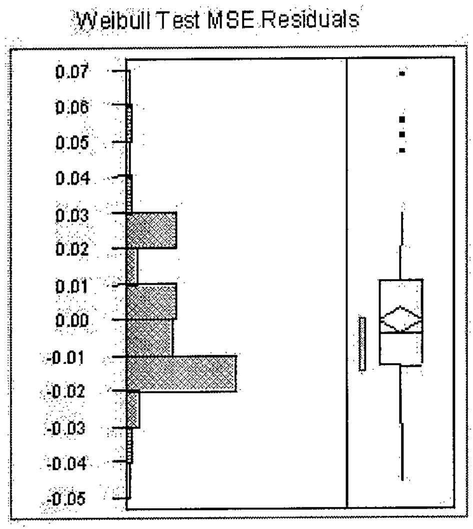


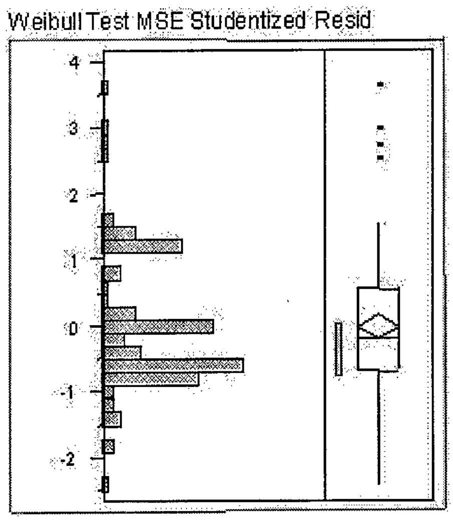
Figure 27. Weibull Pruning: Time Sequence of Residuals for Test MSE

outliers for the size of the sample. For 120 responses, one or two outliers are expected, which Figure 28 depicts. Since the F-test is robust against non-normality (as long as the underlying distribution is not highly skewed) the normality assumption is recognized in continuing with the ANOVA.

Within the five measured responses, initial MSE is a control response to show that the various algorithms start with equal error. For the two levels of Weibull Pruning, the results show no difference in the trained networks' initial MSE at $\alpha = 0.05$. This implies that both levels of Weibull Pruning start with an equivalent network. This is expected since the common data set is surveyed by the initialization routine to establish initial weights prior to training. From that starting point, the training program is consistent in obtaining a network with a common initial MSE.

The ANOVA in Table 4 reveals that the final MSE for the training and test data sets are not affected by the two levels of Weibull Pruning, with p-values of 0.7939 and





```
   Test for Normality
  Shapiro-Wilk W Test
         W        Prob<W
0.914667          <.0001
```

```
Quantiles
      maximum      100.0%        3.6788
                    97.5%        2.7680
                    90.0%        1.3581
      quartile      75.0%        0.5630
      median        50.0%       -0.1849
      quartile      25.0%       -0.6779
                    10.0%       -0.7931
                     2.5%       -1.7803
      minimum        0.0%       -2.4220
Moments
    Mean                        -0.0054
    Std Dev                      1.0046
    Std Error Mean               0.0917
```

Figure 28. Weibull Pruning: Distribution of Residuals for Test MSE

0.8911, respectively. The trainability and predictability of the network is mainly affected by the size of the initial network architecture. In both cases the interaction between the number of hidden nodes and number of inputs of the initial network is the dominant term of the ANOVA. The significance of the network architecture has already been discussed in Chapter II with regards to a network's ability to *learn* and predict. Thus, this result is not surprising. The ANOVA in Table 4 also reveals that percentage of Weibull Pruning has little effect on the number of weights pruned, with a p−value = 0.6078. Again, the model

Training MSE

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	0.00005758	1.0450	0.3089
IN	1	1	0.00010884	1.9753	0.1627
HN*IN	1	1	0.00047861	8.6857	0.0039
Percent Prune	1	1	0.00000378	0.0686	0.7939
HN*Percent Prune	1	1	0.00001064	0.1931	0.6612
IN*Percent Prune	1	1	0.00003030	0.5500	0.4599
HN*IN*Percent Prune	1	1	0.00008641	1.5680	0.2131

Test MSE

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	0.00015540	0.4048	0.5259
IN	1	1	0.00068091	1.7736	0.1856
HN*IN	1	1	0.00151631	3.9496	0.0493
Percent Prune	1	1	0.00000722	0.0188	0.8911
HN*Percent Prune	1	1	0.00001786	0.0465	0.8296
IN*Percent Prune	1	1	0.00005949	0.1549	0.6946
HN*IN*Percent Prune	1	1	0.00015365	0.4002	0.5283

Weights Pruned

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	5.3088	0.1451	0.7040
IN	1	1	1.3886	0.0380	0.8459
HN*IN	1	1	2620.8800	71.6489	<.0001
Percent Prune	1	1	9.6875	0.2648	0.6078
HN*Percent Prune	1	1	11.3882	0.3113	0.5780
IN*Percent Prune	1	1	16.8583	0.4609	0.4986
HN*IN*Percent Prune	1	1	33.8000	0.9240	0.3385

Table 4. ANOVA of Resulting MSEs and Prune Count for Two Weibull Levels

indicates that interaction between the initial number of inputs and hidden nodes plays the only significant role in the number of weights eventually pruned.

The two pruning levels are not significant at $\alpha = 0.05$ in the final network's performance or complexity. However, the ANOVA in Table 5 reveals the measure of floating point operations or efficiency of pruning does respond to the interaction between all three

Floating Point Operations

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	140.2547	6.5714	0.0117
IN	1	1	370.4572	17.3571	<.0001
HN*IN	1	1	1363.1441	63.8677	<.0001
Percent Prune	1	1	10.0474	0.4708	0.4941
HN*Percent Prune	1	1	29.1396	1.3653	0.2451
IN*Percent Prune	1	1	87.5372	4.1014	0.0452
HN*IN*Percent Prune	1	1	259.7821	12.1716	0.0007

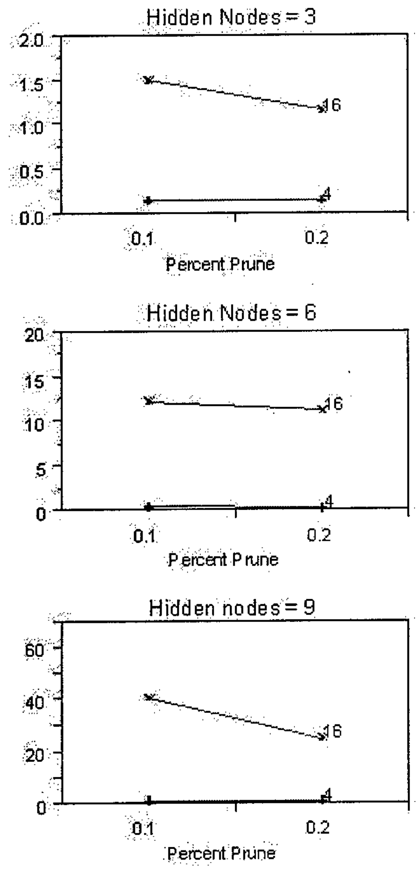
Table 5. ANOVA of Floating Point Operations for Two Weibull Levels

factors: number of inputs, hidden nodes, and CDF level for the percentage pruned in an iteration.

The CDF level is within the dominant interaction, so Figure 29 illustrates the conditional effect the number of inputs and CDF level have on floating point operations at a given number of hidden nodes. For 3 or 9 hidden nodes, the interaction is evident and to lesser extent for 6 hidden nodes, with a decreasing computational burden for the higher level of inputs and pruning. Likewise, Figure 30 shows an evident interaction conditioned on the number of inputs. The efficiency improves for 20% Weibull Pruning, especially as the initial complexity of the network increases.

To further illustrate the difference in efficiency, the paired t-tests for the two levels of Weibull Pruning (family significance level of $\alpha = 0.05$ for six Bonferroni comparisons) are shown in Table 6. The paired t-tests reveal that across the six combinations of the number of inputs and hidden nodes, Weibull Pruning at a CDF level of 20% requires significantly fewer floating point operations than at the 10% level for all but the 16 input, 6 hidden node network. In that case, the difference is in favor of 20% pruning, but it is not significant (recall Figures 29 and 30 show the least interaction for those respective levels).

The scale of floating point operations increases with the initial network size. Larger networks may require more pruning, which benefits from a more aggressive pruning percentage. So as networks increase in size, the level of Weibull Pruning plays an important role in the efficiency of network reduction. The 20% Weibull Pruning approach becomes the benchmark for further comparisons.



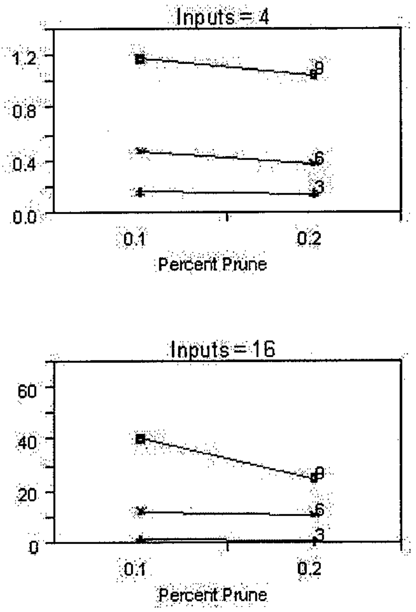


Figure 29. Interaction of 10% and 20% Weibull Pruning with 4 and 16 Inputs on Floating Point Operations, Conditioned on Hidden Nodes

Figure 30. Interaction of 10% and 20% Weibull Pruning with 3, 6, and 9 Hidden Nodes on Floating Point Operations, Conditioned on Inputs

Hidden Units	Inputs	Mean Difference	t-Ratio	Prob $>\|t\|$
3	4	0.00831	6.8419	$< .0001$
3	16	0.35097	5.9147	0.0002
6	4	0.09890	9.5458	$< .0001$
6	16	1.12926	2.9464	0.0163
9	4	0.12151	6.0375	0.0002
9	16	14.88033	5.8237	0.0003

Table 6. Weibull Pruning at 10% Vs 20%: Paired t-tests of Floating Point Operations for Various Network Stuctures (10% Weibull - 20% Weibull)

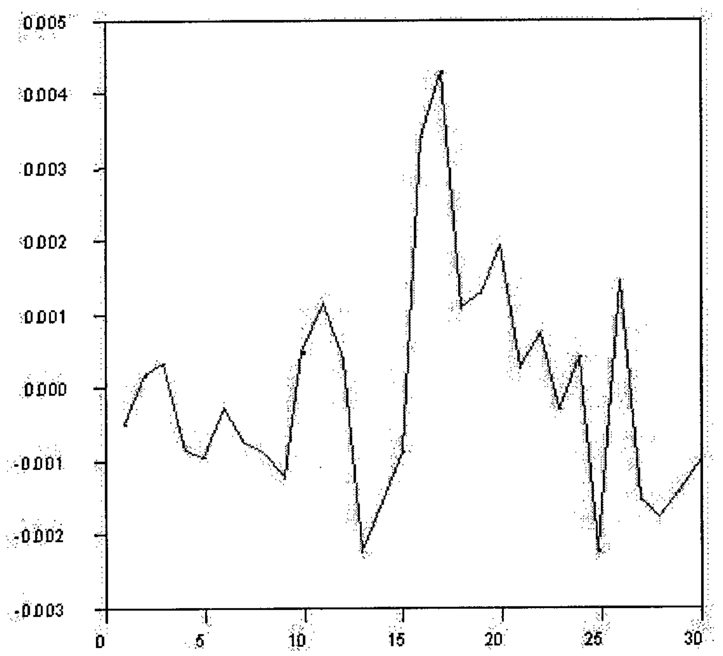
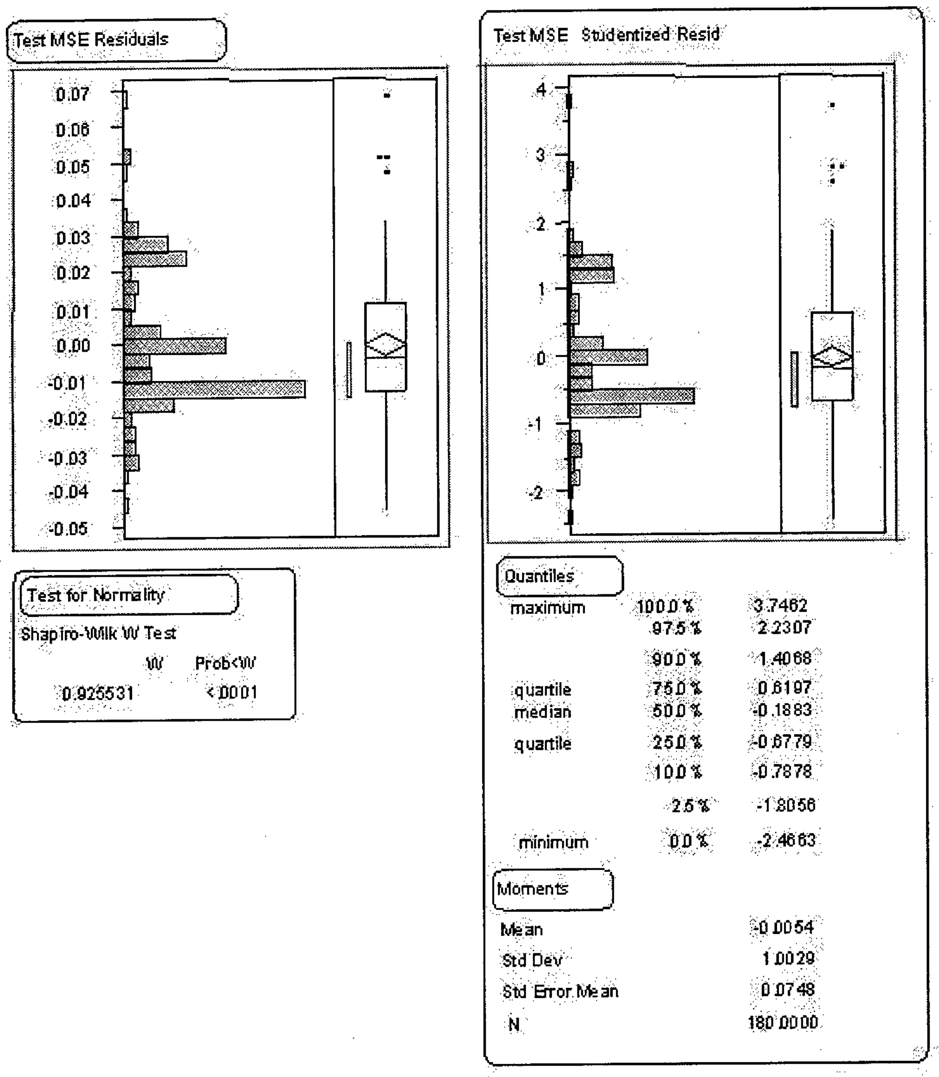


Figure 31. Three Pruning Algorithms: Time Sequence of Residuals for Test MSE

4.3.2 Competing Pruning Algorithms. Tukey-Kramer MCP decides which weights to prune based on the least salience and the correlated saliences not significantly different. Tukey-Kramer MCP uses B bootstrap samples in calculating mean saliences. The sampling requirement can be counterproductive if the network is not large enough to yield an adequate subset of weights to eliminate. However, of equal concern is whether Tukey-Kramer MCP compares in final network performance with other pruning algorithms.

The salience distribution has a Pareto appearance, but not the fit. The saliences do fit a Weibull distribution; and so, Weibull Pruning uses the single sample's distribution characteristics to eliminate the trivial many saliences. The efficiency of Weibull Pruning at no cost to final network performance is subject to comparison. Chapter III justifies OBS as the classical pruning algorithm for comparison.

The simulation of Table 3 with three levels of hidden nodes and two levels of inputs is analyzed for OBS, Tukey-Kramer MCP, and the Weibull level chosen in Section 4.3.1. For the 10 replications of each network structure, responses are recorded upon the completion of each pruning algorithm. The independent machines merge the response data to a single file for analysis. Again, the responses from any given machine can be time sequenced, but not the total 180 values for any given response. Figure 31 presents a typical random



Test for Normality

Shapiro-Wilk W Test

W	Prob<W
0.925531	<.0001

Quantiles

maximum	100.0 %	3.7462
	97.5 %	2.2307
	90.0 %	1.4068
quartile	75.0 %	0.6197
median	50.0 %	-0.1883
quartile	25.0 %	-0.6779
	10.0 %	-0.7878
	2.5 %	-1.8056
minimum	0.0 %	-2.4663

Moments

Mean	-0.0054
Std Dev	1.0029
Std Error Mean	0.0748
N	180.0000

Figure 32. Three Pruning Algorithms: Distribution of Residuals for Test MSE

pattern found for the ANOVA residuals from the simulation experiment. In this figure, the final test MSE is the response. As before, the analysis suggests an independence of replications within the treatment.

On checking the assumption of normality, a typical residual pattern is shown in Figure 32. In this figure, the final test MSE is the response. Although the Shapiro-Wilk W test again rejects the null hypothesis for normality, the residuals still have a symmetric normal-like distribution about zero. The studentized residuals display a couple of outliers

TrainMSE

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	0.00035951	6.8619	0.0096
IN	1	1	0.00051147	9.7624	0.0021
HN*IN	1	1	0.00304277	58.0769	<.0001
Algorithm	2	2	0.00001060	0.1011	0.9039
HN*Algorithm	2	2	0.00002181	0.2081	0.8123
IN*Algorithm	2	2	0.00005450	0.5201	0.5954
HN*IN*Algorithm	2	2	0.00019876	1.8969	0.1532

TestMSE

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	0.00384472	10.3845	0.0015
IN	1	1	0.01601701	43.2614	<.0001
HN*IN	1	1	0.03654700	98.7123	<.0001
Algorithm	2	2	0.00001583	0.0214	0.9789
HN*Algorithm	2	2	0.00002739	0.0370	0.9637
IN*Algorithm	2	2	0.00008249	0.1114	0.8946
HN*IN*Algorithm	2	2	0.00025648	0.3464	0.7078

Weights Pruned

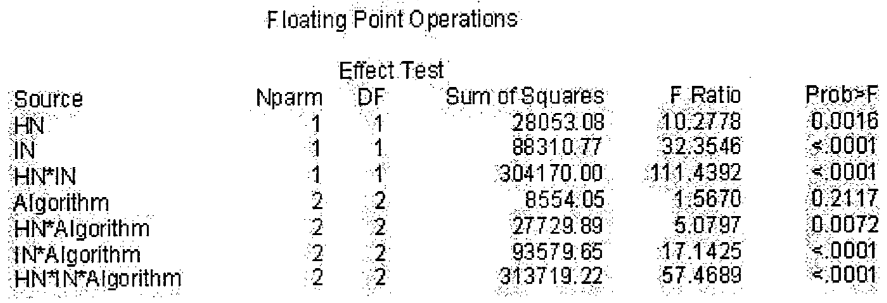
Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	392.832	12.3347	0.0006
IN	1	1	86.429	2.7138	0.1014
HN*IN	1	1	30337.200	952.5736	<.0001
Algorithm	2	2	20.876	0.3277	0.7210
HN*Algorithm	2	2	17.388	0.2730	0.7614
IN*Algorithm	2	2	19.557	0.3070	0.7360
HN*IN*Algorithm	2	2	24.050	0.3776	0.6861

Table 7. ANOVA of Resulting MSEs and Prune Count for Three Pruning Methods

for the sample size of 180, as is expected. Given the robustness of the F-test as discussed earlier, the normality assumption is recognized in continuing with the ANOVA.

The ANOVA in Table 7 reveals that the final MSE for the training and test data sets are not affected by the type of pruning algorithm, with p-values of 0.9039 and 0.9789, respectively. The trainability and predictability of the network is mainly affected by the



Floating Point Operations

Effect Test

Source	Nparm	DF	Sum of Squares	F Ratio	Prob>F
HN	1	1	28053.08	10.2778	0.0016
IN	1	1	88310.77	32.3546	<.0001
HN*IN	1	1	304170.00	111.4392	<.0001
Algorithm	2	2	8554.05	1.5670	0.2117
HN*Algorithm	2	2	27729.89	5.0797	0.0072
IN*Algorithm	2	2	93579.65	17.1425	<.0001
HN*IN*Algorithm	2	2	313719.22	57.4689	<.0001

Table 8. ANOVA of Floating Point Operations for Three Pruning Methods

size of the initial network architecture. In both cases the interaction between the number of hidden nodes and number of inputs of the initial network is the dominant term of the ANOVA. The ANOVA in Table 7 also reveals that the type of pruning has little effect on the number of weights pruned, with a $p-$value $= 0.7210$. Again, the model indicates that interaction between the initial number of inputs and hidden nodes plays the only significant role in the number of weights eventually pruned. The results are consistent with those found in the earlier simulation for the two Weibull CDF levels of pruning.

The type of pruning algorithm is not significant at $\alpha = 0.05$ in the final network's performance or complexity. However, the ANOVA of Table 8 reveals the amount of floating point operations or efficiency of pruning does respond to the interaction between all three factors: number of inputs, hidden nodes, and type of pruning algorithm.

The pruning algorithm type is within the significant three-way interaction, so Figure 33 illustrates the conditional effect the number of inputs and algorithms have on floating point operations at a given number of hidden nodes. For all hidden node conditions, the interaction is evident, with a decreasing computational burden for Weibull Pruning at the higher level of inputs. Likewise, Figure 34 shows an evident interaction conditioned on the number of inputs. The efficiency improves for Weibull Pruning, especially as the initial complexity of the network increases.

To further illustrate the difference in efficiency, the paired t-tests between the three pruning algorithms (family significance level of $\alpha = 0.05$ for eighteen Bonferroni compar-

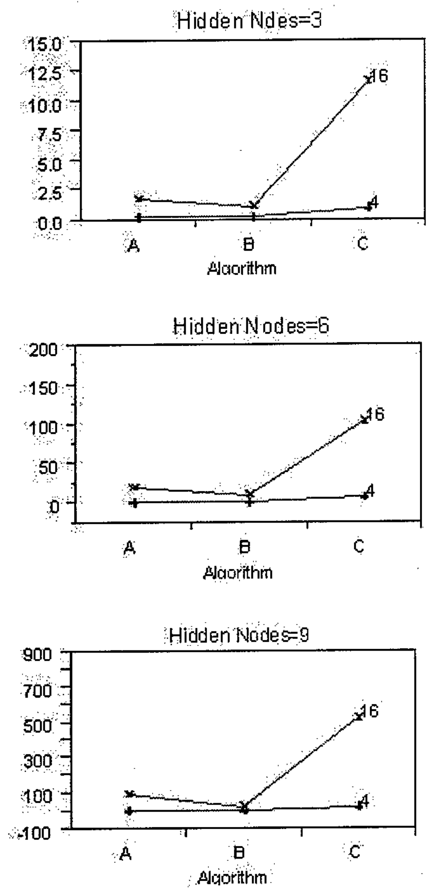


Figure 33. Interaction of OBS(A), Weibull Pruning(B), TK MCP(C) with 4 and 16 Inputs on Floating Point Operations, Conditioned on Hidden Nodes

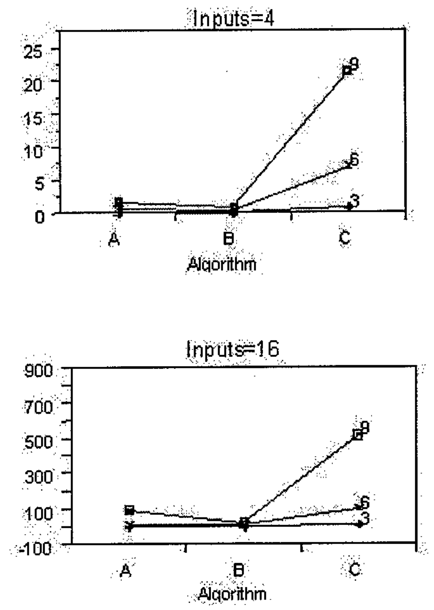


Figure 34. Interaction of OBS(A), Weibull Pruning(B), TK MCP(C) with 3, 6, and 9 Hidden Nodes on Floating Point Operations, Conditioned on Inputs

isons) are performed similar to those in Table 6. The paired t-tests reveal that across the six combinations of the number of inputs and hidden nodes, Weibull Pruning at a CDF level of 20% requires significantly fewer floating point operations than either OBS or Tukey-Kramer MCP for all six network structures. Also, OBS requires significantly fewer floating point operations than Tukey-Kramer MCP for all six network structures. All eighteen paired t-tests have a $p-$value < 0.0001, further supporting the conclusions of the ANOVA: Weibull Pruning is significantly more efficient.

4.3.3 Summary of Results. The first simulation reveals that the CDF level for Weibull Pruning does not affect the final network's performance nor complexity. However, computational burden is significantly reduced (at $\alpha = 0.05$) for the higher CDF level of pruning. The results suggest continuing to prune at the higher CDF level.

The results of the experiments are also encouraging for multiple comparison pruning versus traditional iterative pruning. In comparison to OBS, Tukey-Kramer MCP and Weibull Pruning have no effect on the pruned network's performance. In contrast, the efficiency of pruning is significantly affected: Weibull Pruning has the advantage over both OBS and Tukey-Kramer MCP with significantly fewer floating point operations (at $\alpha = 0.05$). Tukey-Kramer MCP efficiency is dependent on the bootstrap sampling requirement that Weibull Pruning is able to avoid. However, for the computational price, Tukey-Kramer provides insight into the trivial saliences and the corresponding weights subject to removal. Chapter V provides conclusions and recommendations for multiple comparison pruning.

V. Recommendations and Conclusions

5.1 Introduction

Large neural networks have the advantage of quick learning, but too many connections may store the idiosyncrasies of that particular data set. The neural network's ability to predict future examples is hindered by the possible memorization of the training data set, such that the network may become unable to generalize data outside the set. Network pruning should be a judicious reduction in network size so that both training and test error are minimal. The computational burden of pruning algorithms is of concern in this research. Several approaches are devised to address more than one weight or one network model at a time. In order to remove more than one connection, the decision requires information on the relevance of all the connections. Statistical inference procedures are the tool to allow subset elimination of connections. In this chapter, contributions advancing the process of neural network pruning are summarized and recommendations for future research are made.

5.2 Summary

This research begins with a review of neural network construction, salience measures of the weights within the networks, and statistical inference techniques that may apply to pruning a more efficient, better generalizing network. A review of neural network pruning details the computational intensity to produce the salience measures necessary to decide which weights to eliminate. The significant contribution of this research is the introduction of new concepts and methods in the field of multiple weight elimination of neural networks. A summary of research advances and contributions in this area of increasing the efficiency of neural network pruning follows.

5.2.1 Review of Tukey-Kramer Multiple Comparison Pruning. A common theme amongst pruning algorithms is the role of salience measures in determining which weights are important. Weights with small saliences have little affect on error and are subject to pruning. However, sometimes obtaining saliences may be computationally burdensome. The efficiency of weight elimination becomes an issue for large networks posing a formidable task. Given multiple saliences for the weights in question, multiple comparisons improve the decision making steps within the pruning algorithm. Rather than eliminate weights one at a time, comparatively small saliences are grouped and eliminated in batch. Batch elimination reduce the number of loops, retraining, or Hessian matrix inversions in a pruning algorithm.

The Tukey-Kramer MCP considers the set of all pairwise comparisons of the hypotheses in Equation 6, while accounting for the correlation between sample means. The simultaneous confidence interval in Equation 8 identifies the subset with the least salience given heteroscedastic, correlated components. The resulting subset of saliences identify the weights to be eliminated.

Results favorably indicate that Tukey-Kramer MCP has no detrimental effect on the final performance of the neural network versus the traditional OBS pruning approach. Tukey-Kramer MCP requires fewer pruning cycles than OBS to reach the stopping criterion. However in this research, the network size works against the Tukey-Kramer MCP resampling requirement for mean comparisons. In order to obtain mean saliences, Tukey-Kramer MCP can require significantly more computations than OBS.

5.2.2 Review of Weibull Pruning. For a large neural network, saliences have a precipitous decline from the left to suggest a trivial many exist in conjunction with a vital few. The Pareto-like drop is best described by a Weibull distribution, from the results of a goodness-of-fit test. The Weibull distribution uses the data to select the distribution and fit the parameters, avoiding the Tukey-Kramer MCP resampling requirement.

The implementation of Weibull Pruning is a modification of existing pruning algorithms, where the Weibull distribution supports the decision to remove multiple connections from the network. A goodness-of-fit test is part of the algorithm decision prior to pruning a subset of weights. If the Weibull distribution is rejected, the algorithm performs a single weight elimination. Likewise, the notion of trivial many saliences dictates subset pruning only when the shape estimator, $\hat{\beta}$, is less than one (see Figure 24). The MLEs of Equations 17 and 18 are used in the confidence interval of Equation 19 to group the trivial many saliences. Their weights are eliminated and the rest of the weights are updated based on the optimal weight change prescribed by the largest salience within the subset eliminated.

Provided $\hat{\beta}$ is less than one, Weibull Pruning is an appropriate alternative to the Tukey-Kramer MCP. Results favorably indicate Weibull Pruning has no detrimental effect on the final performance of the neural network versus Tukey-Kramer MCP or the traditional OBS pruning approach. Weibull Pruning requires fewer pruning cycles than OBS to reach the stopping criterion. More importantly, Weibull Pruning requires significantly fewer computations than OBS and Tukey-Kramer MCP at $\alpha = 0.05$.

5.3 Recommendations

There are related research topics that could not be covered within the scope of this research effort. Two of the research topics could be pursued with worthwhile benefits.

The first research topic is the application of Tukey-Kramer MCP and Weibull Pruning to ongoing neural network pruning research. OBS is a benchmark approach, but work continues in improving neural network capability to better generalize. Applying subset elimination of weights through statistical inference may improve any research effort underway.

The second research topic is the utility of Tukey-Kramer MCP on larger neural networks. Table 2 suggests that larger networks may benefit from Tukey-Kramer MCP in computational efficiency when the number of candidate weights for elimination offset the resampling burden. Hybrid pruning algorithms may use Tukey-Kramer MCP to judiciously prune the trivial many weights when the potential for a large subset is high. Another approach may consider an earlier stopping criterion when using Tukey-Kramer MCP to avoid counterproductive resampling when the potential for subset elimination is low.

Bibliography

1. Abernethy, R.B. *New Weibull Handbook*. Houston TX: Gulf Publishing Co, 1996.

2. Abu-Mostafa, Y.S. "The VC Dimension: Information Versus Complexity in Learning," *Neural Computation*, *1*:312–317 (1989).

3. Akaike, H. "A New Look at Statistical Model Identification," *IEEE Transactions on Automatic Control*, *19*:716–723 (1974).

4. Anderson, J.A. and E. Rosenfeld (eds). *Neurocomputing: foundations of research.* Cambridge MA: MIT Press, 1988.

5. Arnold, B.C. *Pareto Distributions*. Fairland MD: International Co-operative Publishing House, 1983.

6. Baum, E.B. and D. Haussler. "What Size Net Gives Good Generalization?," *Neural Computation*, *1*:151–160 (1989).

7. Baxt, W.G. and H. White. "Bootstrapping Confidence Intervals for Clinincal Input Variable Effects in a Network Trained to Identify the Presence of Acute Myocardial Infarction," *Neural Computation*, *7*:624–638 (1995).

8. Baxter, M.A. "Minimum Variance Unbiased Estimation of the Parameters of the Pareto Distribution," *Metrika*, *27*:133–138 (1980).

9. Bishop, Christopher M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

10. Bishop, Christopher M. and Michael I. Jordan. *Neural Networks*. Technical Report AI Memo No. 1562, Artificial Intelligence Laboratory: Massachusetts Insitute of Technology, March 1996.

11. Brown, L.D. "A Note on the Tukey-Kramer Procedure for Pairwise Comparisons of Correlated Means." *Design of Experiments: Ranking and Selection56*, edited by T.J. Santner and A.C. Tamhane. 1–6. Marcel Dekker, Inc, 1984.

12. Cherkassky, V. "Preface." *From Statistics to Neural Networks: Theory and Pattern Recognition Applications* edited by J.H. Cherkassky, V. Friedman and H. Wechsler, vi–vii, Berlin: Springer-Verlag, 1994.

13. Demuth, H. and M. Beale. *Neural Network Toolbox for Use with Matlab*. Natick MA: The MathWorks Inc, 1994.

14. Dudewicz, E.J. and T.A. Bishop. "The Heteroscedastic Method." *Optimizing Methods in Statistics*. 183–203. New York: Academic Press, 1979.

15. Dudewicz, E.J. and T.A. Bishop. "Heteroscedastic ANOVA," *Sankhya*, *43*(B):40–57 (1981).

16. Dudewicz, E.J.; T.W. Lin and C.W. Swenson. "Stronger Tests Using Heteroscedastic ANOVA with Simulations of Accounting Systems," *American Journal of Mathematical and Management Sciences*, *12*(4):277–299 (1992).

17. Dunn, O.J. "Confidence Intervals for the Means of Dependent Normally Distributed Variables," *Journal of the American Statistical Association*, *54*:613–621 (1959).

18. Efron, B. *The Jackknife, the Bootstrap and Other Resampling Plans*. Philadelphia: SIAM, 1982.

19. Friedman, M. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance," *Journal of the American Statistical Association*, *32*:675–701 (1937).

20. Gibbons, J.D.; I. Olkin and M. Sobel. *Selecting and Ordering Populations: A New Statistical Methodology*. John Wiley and Sons, 1977.

21. Golden, R.M. "Making Correct Statistical Inferences Using the Wrong Probability Model," *Journal of Mathematical Psychology*, *39*:3–20 (1995).

22. Golden, R.M. *Mathematical Methods for Neural Network Analysis and Design*. MIT Press, 1996.

23. Gorodkin, J.; L.K. Hansen; A. Krogh; C. Svarer and O. Winther. "A Quantitative Study of Pruning by Optimal Brain Damage," *International Journal of Neural Systems*, *4*(2):159–169 (1993).

24. Harter, H.L. *Order Statistics and Their Use in Testing and Estimation, Vol. 1*. Aerospace Research Laboratories, Office of Aerospace Research, U.S. Air Force, 1969.

25. Hassibi, B. and D.G. Stork. "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon." *Advances in Neural Information Processing Systems V*. 164–171. San Mateo CA: Morgan Kaufmann, 1993.

26. Hassibi, B.; D.G. Stork and G. Wolff. "Optimal Brain Surgeon: Extensions and Performance Comparisons." *Advances in Neural Information Processing Systems VI*. 263–270. San Mateo CA: Morgan Kaufmann, 1994.

27. Hertz, J.; A. Krogh and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, 1991.

28. Hochberg, Y. and A.C. Tamhane. *Multiple Comparison Procedures*. John Wiley and Sons, 1987.

29. Hsu, J.C. *Multiple Comparison: Theory and Methods*. Chapman and Hall, 1996.

30. Kang, S.B. and Y.S. Cho. "Estimation of the Parameters in a Pareto Distribution by Jackknife and Bootstrap Method," *Journal of Information and Optimization Sciences*, *18(2)*:289–300 (1997).

31. Kramer, C.Y.. "Extension of Multiple Range Tests to Group Correlated Adjusted Means," *Biometrics*, *13*:13–18 (1957).

32. Law, A.M. and S. Vincent. *ExpertFit, Version 1.50*. Tucson AZ: Averill M. Law and Associates, 1998.

33. Le Cun, Y.; J.S. Denker and S.A. Solla. "Optimal Brain Damage." *Advances in Neural Information Processing Systems II*. 598–605. San Mateo CA: Morgan Kaufmann, 1990.

34. Lehman, A. and J. Sall. *JMP Statistics and Graphics Guide, Version 3.1*. Cary NC: SAS Institute Inc, 1995.

35. Levin, A.U.; T.K. Leen and J.E. Moody. "Fast Pruning Using Principal Components." *Advances in Neural Information Processing Systems VI*. 35–42. San Mateo CA: Morgan Kaufmann, 1994.

36. Malik, H.J. "Estimation of the Parameters of the Pareto Distribution," *Metrika*, *15*:126–132 (1970).

37. Moore, A.H. and H.L. Harter. "One-Order-Statistic Conditional Estimators of Shape Parameters of Limited and Pareto Distributions and Scale Parameters of Type II Asymptotic Distributions of Smallest and Largest Type," *IEEE Transactions on Reliability*, *R-16*:100–103 (1967).

38. Mozer, M.C. and P. Smolensky. "Skeletonization: A Technique for Trimming the Fat from a Network Via Relevance Assessment." *Advances in Neural Information Processing Systems I*. 107–115. San Mateo CA: Morgan Kaufmann, 1989.

39. Muniruzzaman, A.N.M. "On Measures of Location and Dispersion and Tests of Hypotheses in a Pareto Population," *Bulletin of Calcutta Statistical Association*, *7*:115–123 (1993).

40. Neter, J.; M.H. Kutner; C.J. Nachtsheim and W. Wasserman. *Applied Linear Statistical Models*. Chicago IL: Irwin, 1996.

41. Paass, G. "Assessing and Improving Neural Network Predictions by the Bootstrap Algorithm." *Advances in Neural Information Processing Systems V* 196–203, San Mateo CA: Morgan Kaufmann, 1993.

42. Pareto, V. *Cours d'Economie Politique*. Lausanne, 1897.

43. Quandt, R.E. "Old and New Methods of Estimation and the Pareto Distribution," *Metrika*, *10*:55–82 (1966).

44. Reed, R. "Pruning Algorithms - A Survey," *IEEE Transactions on Neural Networks*, *4*(5):740–747 (1993).

45. Ripley, B.D. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

46. Rosenblatt, F. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, *65*:386–408 (1958). Reprinted in Anderson and Rosenfeld (1988).

47. Sachs, L. *Applied Statistics: A Handbook of Techniques*. Springer-Verlag, 1984.

48. Saksena, S.K. *Estimation of Parameters in a Pareto Distribution and Simultaneous Comparison of Estimators*. PhD dissertation, Louisiana Tech University, Ruston LA, March 1978.

49. Saksena, S.K. and A.M. Johnson. "Best Unbiased Estimators for the Parameters of a Two-Parameter Pareto Distribution," *Metrika*, *31*:77–83 (1984).

50. Sarle, W.S. "Neural Networks and Statistical Models." *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. 1994.

51. Setiono, R. "A Penalty Function Approach for Pruning Feedforward Neural Networks," *Neural Computation*, *9*:185–204 (1997).

52. Stein, C. "A Two-Sample Test for a Linear Hypothesis Whose Power is Independent of the Variance," *Annals of Mathematical Statistics*, *16*:243–258 (1945).

53. Steppe, J.M. and K.W. Bauer Jr. "Improved Feature Screening in Feedforward Neural Networks," *Neurocomputing*, *13*:47–58 (1996).

54. Svarer, C.; L.K. Hansen; J. Larsen and C.E. Rasmussen. "Designer Networks for Time Series Processing." *Neural Networks for Signal Processing 3*, edited by C.A. Kamm, 78–87, IEEE, 1993.

55. Thrun, S.B.; J. Wnek; R.S. Michalski; T. Mitchell; J. Cheng, et al. *The Monk's Problems — A Performance Comparison of Different Learning Algorithms*. Technical Report CMU-CS-91-197, Computer Science Department: Carnegie Mellon University, 1991.

56. Tukey, J.W. "Discussion, Emphasizing the Connection between Analysis of Variance and Spectrum Analysis," *Technometrics*, *3*:191–219 (1961).

57. Tukey, J.W. "The Philosophy of Multiple Comparisons," *Statistical Science*, *6(1)*:100–116 (1991).

58. Vapnik, V.N. and A.Y. Chervonenkis. "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities," *Theory of Probability and its Applications*, *16*:264–280 (1971).

59. Vuong, Q.H. "Liklihood Ratio Tests for Model Selection and Non-nested Hypotheses," *Econometrica*, *57*:307–333 (1989).

60. Wald, A. "Tests of Statistical Hypotheses Concerning Several Parameters When the Number of Observations is Large," *Transactions of the American Mathematical Society*, *54*:426–482 (1943).

61. Wang, C.; S.S. Venkatesh and J.S. Judd. "Optimal Stopping and Effective Machine Complexity in Learning." *Advances in Neural Information Processing Systems VI*. 303–310. San Mateo CA: Morgan Kaufmann, 1994.

62. White, H. "Learning in Artificial Neural Networks: A Statistical Perspective," *Neural Computation*, *1*:425–464 (1989).

63. Widrow, B. and M.E. Hoff. "Adaptive Switching Circuits," *IRE WESCON Convention Record*, *4*:94–104 (1960). Reprinted in Anderson and Rosenfeld (1988).

64. Wilks, S.S. "The Large Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses," *Annals of Mathematical Statistics*, *9*:60–62 (1938).

Vita

Major Donald E. Duckro was born on 27 April 1962 in Dayton, Ohio. In 1980, he graduated from Carroll High School and attended the University of Dayton in Ohio. In 1984, he graduated from the University of Dayton with a Bachelor of Chemical Engineering. His first assignment in the United States Air Force was in an Undergraduate Engineering Conversion Program at Louisiana Tech University in Ruston, Louisiana. He graduated with a Bachelor of Science Degree in Electrical Engineering in 1986. His follow-on assignment at Los Angeles AFB, California was as a satellite development engineer for the Global Positioning System Joint Program Office. Major Duckro returned to the University of Dayton in 1989 to study for a Master's of Science in Applied Mathematical Sciences. In 1990, he graduated and joined the faculty of the United States Air Force Academy. He departed the academy in 1994 as an assistant professor, and returned to Dayton as a program manager for the C-130J Hercules tactical air lifter at Wright-Patterson AFB, Ohio. Upon winning a Dayton Area Graduate Studies Institute competitive scholarship in 1995, he began class work for a doctoral program. He entered the doctoral program full-time in 1996 with the Air Force Institute of Technology as the degree granting institution.

Permanent address: 2313 Revere Avenue
Dayton, OH 45420

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 99	3. REPORT TYPE AND DATES COVERED Dissertation

4. TITLE AND SUBTITLE Multiple Comparison Pruning of Neural Networks	5. FUNDING NUMBERS
6. AUTHOR(S) Donald E. Duckro, Major, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Bldg 640 2950 P Street Wright-Patterson AFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/DS/ENC/99-02
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dayton Area Graduate Studies Institute 3155 Research Boulevard Suite 205 Kettering OH 45420 (937) 781-4000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Advisor: Dr. Dennis W. Quinn, ENC

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Reducing a neural network's complexity improves the ability of the network to be applied to future examples. Like an overfitted regression function, neural networks may miss their target because of the excessive degrees of freedom stored up in unnecessary parameters. Over the past decade, the subject of pruning networks has produced non-statistical algorithms like Skeletonization, Optimal Brain Damage, and Optimal Brain Surgery as methods to remove connections with the least salience. There are conflicting views as to whether more than one parameter can be removed at a time. The methods proposed in this research use statistical multiple comparison procedures to remove multiple parameters in the model when no significant difference exists. While computationally intensive, Tukey-Kramer Multiple Comparison Pruning compares well with Optimal Brain Surgery in pruning and network performance. When the Tukey-Kramer method has inefficient sampling requirements, Weibull distribution theory alleviates the computational burden of bootstrap resampling with single sample analysis, maintaining comparable network performance.

14. SUBJECT TERMS Backpropagation, Bootstrap, Subset selection, Tukey-Kramer Multiple Comparison Procedure, Weibull distribution	15. NUMBER OF PAGES 86
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL